Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Titulació:

Enginyeria Automàtica i Electrònica Industrial

Alumne:

Joan Pozo Prades

Títol PFC:

Estudi d'un sistema de control amb aprenentatge per reforç

Director del PFC:

Bernardo Morcego Seix

Convocatòria de lliurament del PFC

Gener de 2012

Contingut d'aquest volum: **-MEMÒRIA-**

ESTUDI D'UN SISTEMA DE CONTROL AMB APRENENTATGE PER REFORÇ

Autor: Joan Pozo Prades

Tutor: Bernardo Morcego Seix

Convocatòria: Gener de 2012

ÍNDEX MEMÒRIA

ÍNDEX DE FIGURES

CAPÍTOL 1: INTRODUCCIÓ

1.1. Objecte

Estudi d'una tècnica d'aprenentatge per reforç aplicada al disseny de controladors, i de les seves prestacions i característiques d'estabilitat, utilitzant com a proves diferents sistemes a controlar, els quals tenen característiques que els fan difícils de controlar, com la inestabilitat.

1.2. Especificacions

En aquest punt del projecte es descriuran les diferents restriccions que s'han acordat amb el tutor.

- El conjunt de sistemes a tractar seran diferents entre ells.
- El conjunt de sistemes a tractar serà de màxim 4.
- El projecte se centrarà en 1 tècnica d'aprenentatge per reforç.
- A cada sistema s'aplicarà una pertorbació per estudiar l'adaptació del mètode emprat i la capacitat de rebutjar pertorbacions de càrrega.

1.3. Abast

En aquest punt es pretén afitar el projecte, pel que es defineixen els punts que es realitzaran com els que no estaran inclosos en el projecte.

- Realitzar una selecció de sistemes senzills.
- Estudi de diferents sistemes seleccionats a controlar, és a dir, establir característiques del sistema i obtenir un controlador lineal per aquests.
- Estudi de les diferents tècniques d'aprenentatge per reforç trobades a la literatura.
- Realitzar una selecció de la tècnica a utilitzar en el projecte.
- Estudi de la tècnica seleccionada aplicada als sistemes seleccionats.
- Es realitzarà la programació de la tècnica d'aprenentatge per reforç amb Matlab.
- L'estudi final realitzat no s'aplicarà en cap sistema físic real.

1.4. Justificació

Avui en dia s'utilitza els mètodes de control bàsic, PID, en la majoria de sistemes donat que el seu bon funcionament està més que demostrat. Tot i així, hi ha sistemes en els que un control clàssic té dificultats en realitzar la seva tasca per això s'utilitzen altres tècniques de control més complexes.

El que tracta aquest projecte és estudiar l'aplicació d'una tècnica d'intel·ligència artificial per tal de controlar sistemes. Es vol realitzar una anàlisi contraposant aquestes dues tecnologies per veure els resultats i obtenir unes conclusions. Unes conclusions que podrien ser beneficioses pel que fa al camp de la IA, doncs es podria veure com la seva aplicació en l'àmbit del control de sistemes, que podrien

esdevenir màquines, vehicles, etc., és realment útil i pot ser millor que altres opcions utilitzades en l'actualitat.

CAPÍTOL 2: INTRODUCCIÓ A L'APRENENTATGE PER REFORÇ

El contingut del capítol és extret del llibre escrit per Sutton i Barto[1] però de forma resumida, per tant, si es desitja conèixer el tema amb més profunditat, en els capítols 1 i 3 del llibre s'explica de manera més extensa.

2.1. Introducció

La primera idea que ve al cap és que les persones aprenen gràcies a la interacció amb el seu medi. Quan un nadó juga, mou els braços o simplement observa al seu voltant, no té un professor que l'ajudi a entendre i aprendre sinó que s'ajuda directament de la connexió sensor-motor amb el medi. Exercitant i afavorint aquesta connexió produeix una gran quantitat d'informació de causa i efecte, sobre les

conseqüències de les accions, i sobre el que s'ha de fer per aconseguir allò que es desitja tenir.

Durant la resta de la vida aquestes interaccions són la major font de coneixement de l'entorn i de la pròpia persona en si. Mentre s'aprèn a conduir un cotxe o es manté una conversació, s'està atenent en com l'entorn respon al que es fa, i es busca influir en el que succeeix mitjançant el comportament. Aprendre de les interaccions és la base fonamental de gairebé totes les teories d'aprenentatge i intel·ligència.

L'aprenentatge automàtic és la ciència del disseny i desenvolupament d'algoritmes que permet als ordinadors aprendre. Un ordinador aprèn quan és capaç d'evolucionar cap a cert comportament basat en dades empíriques. En general, en l'aprenentatge automàtic es diferencien 3 formes d'aprenentatge:

- Aprenentatge supervisat
- Aprenentatge no supervisat
- Aprenentatge per reforç

Els algoritmes d'aprenentatge supervisat aprenen d'exemples. El comportament que es desitja es mostra i per tant l'agent tracta de copiar aquest comportament. Per tant, l'entrada i la sortida desitjada del sistema són coneguts. Aquest tipus d'aprenentatge només es pot utilitzar si la sortida desitjada és coneguda.

L'aprenentatge no supervisat només utilitza entrades per evolucionar en el seu comportament. Classificar, agrupar i organitzar dades d'entrada o aprendre a estimar certes dimensions són exemples d'aquest tipus d'aprenentatge. L'objectiu dels algoritmes d'aprenentatge no supervisat és trobar una relació entre diferents entrades. Aquesta forma d'aprenentatge és molt útil per interpretar i processar una gran quantitat de dades. Processar i interpretar els sentits d'una persona podria ser vist com una forma d'intel·ligència. El reconeixement facial és un exemple d'aprenentatge no supervisat.

L'aprenentatge per reforç aprèn de recompenses i penalitzacions (recompenses negatives). En alguns casos el comportament o sortida són desconeguts fins i tot quan l'objectiu de l'agent és clar. Aquests algoritmes tracten de trobar un comportament de tal manera que la suma de les recompenses i penalitzacions sigui màxima. No només tenen en compte les recompenses immediates sinó que tenen en compte les futures recompenses per tal d'evolucionar al comportament desitjat.

2.2. Història

En la història de l'aprenentatge per reforç s'hi troben dues branques importants, les quals es van seguir de manera independent abans de convergir en l'aprenentatge per reforç modern.

1. ***Prova i error.*** Va començar en la psicologia de l'aprenentatge animal.
2. ***Problema del control òptim.*** Utilitza funcions de valor i programació dinàmica.

Però hi ha una tercera branca, menys distingida, és la que fa referència a:

3. Mètodes de diferència temporal. (temporal-difference methods)

Aquestes tres branques s'unirien cap a finals de 1980 per crear el camp de l'aprenentatge per reforç.

2.2.1. Prova i error

Va començar en la psicologia, on les teories d'aprenentatge per reforç són comuns. El primer en resumir breument l'essència de l'aprenentatge per prova i error va ser Edward Thorndike al 1911. Ho anomenava "The Law of Effect" (la llei de l'efecte).

La Llei de l'Efecte inclou dos aspectes molt importants del que significa aprenentatge per prova i error.

Seleccionable. Implica provar alternatives i seleccionar-les entre unes quantes comparant les conseqüències.
Associatiu. Les alternatives trobades per selecció són associades a situacions particulars.

A principis de la intel·ligència artificial, molts investigadors van explorar l'aprenentatge prova-error com un principi d'enginyeria.

1949	Donald Hebb	Publica el llibre The Organization of Behavior, en el qual s'estableixen alguns principis del funcionament de les neurones, com ara la Llei de Hebb.
1954	Minsky i Farley i Clark	Minsky va tractar els models computacionals d'aprenentatge per reforç i va descriure la construcció d'una màquina analògica, format per components que anomenava SNARCs (Stochastic Neuronal-Analog Reinforcement Calculators). Farley i Clark va descriure una altra màquina d'aprenentatge amb xarxa neuronal per aprendre el prova-error.
1960		Es pot trobar fàcilment els conceptes "reforç" i "aprenentatge per reforç" a la literatura d'enginyeria.
1960-1970		És difícil trobar investigadors en la recerca de l'aprenentatge prova-error degut a la confusió que hi ha entre l'aprenentatge supervisat i l'aprenentatge per reforç. La confusió va aparèixer en el moment que es va dubtar d'utilitzar situacions conegudes en el propi aprenentatge.
1973	Widrow, Grupta i Maitra	Van modificar l'algoritme de Widrow i Hoff per fer una regla d'aprenentatge per reforç de senyals d'èxit i fracàs en comptes d'utilitzar exemples d'entrenament.
1975	John Holland	Va esbossar una teoria general de sistemes adaptatius basat en principis seleccionables.
1986	John Holland	Va introduir el *sistema classificador*, sistemes d'aprenentatge per reforç incloent associació i funcions valor. Una de les claus d'aquest sistema era l'algoritme genètic.

Molts de treballs recents mostren que l'aprenentatge per reforç i l'aprenentatge supervisat són en realitat bastant diferents. Altres estudis mostren com l'aprenentatge per reforç podria resoldre problemes en l'aprenentatge de xarxes neuronals, en particular, com es podrien produir algoritmes d'aprenentatge per xarxes de múltiples capes.

2.2.2. Control òptim

Finals 1950		Apareix el terme control òptim per descriure el problema de dissenyar un controlador per minimitzar un senyal d'un sistema dinàmic en el temps.
Mitjans 1950	Richard Bellman i companys	Realitza una aproximació al problema. Aquesta aproximació utilitza el concepte estat dinàmic d'un sistema i d'una funció valor, o "funció de retorn òptim".
1957	Richard Bellman	Els mètodes per resoldre problemes de control òptim es coneixeran per programació dinàmica.
1957	Richard Bellman	Introdueix la versió estocàstica discreta coneguda com MDP (Markovian Decision Processes).
1960	Ron Howard	Va idear el mètode iteratiu pel MDPs.

La programació dinàmica és considerada l'única manera fiable de resoldre problemes generals de control òptim estocàstic. Pateix del que Bellman anomenava "la maledicció de la dimensionalitat", significa que els requeriments computacionals van creixent exponencialment amb el nombre de variables d'estat. La programació dinàmica s'ha estès i desenvolupat al llarg de les últimes dècades, incloent mètodes d'aproximació i mètodes asíncrons.

2.2.3. Aprenentatge diferència temporal

Aquests mètodes són diferents doncs van en funció de la diferència entre les estimacions successives en el temps.

Els orígens de l'aprenentatge en diferència temporal es deu en part a l'aprenentatge psicològic animal, concretament en la noció dels reforçadors secundaris. Un reforçador secundari és un estímul que s'ha vinculat amb un reforçador primari com el menjar o dolor i, com a resultat, ha obtingut les propietats similars al reforç.

1954	Minsky	Seria el primer en adonar-se que el principi dels reforçadors secundaris en la psicologia podria ser important per l'aprenentatge artificial
1959	Arthur Samuel	Va ser el primer en proposar i implementar un mètode d'aprenentatge que contenia les idees de la diferència temporal.
1972	Kplof	Utilitzaria components importants de l'aprenentatge en diferència temporal a l'aprenentatge prova-error. Va desenvolupar la idea del "reforçament generalitzat", on cada component veia totes les entrades en termes de reforç, entrades excitatòries com a recompensa i entrades inhibidores com a penalitzacions.
1981	Sutton	Desenvolupa un mètode per aprenentatge en diferència temporal per l'aprenentatge prova-error, conegut com arquitectura actor-crític.
1988	Sutton	Separa l'aprenentatge en diferència temporal del control, tractant-ho com un mètode de predicció general. Va introduir l'algoritme TD(λ)

Finalment, les branques de diferència temporal i control òptim van convergir en una al 1989 de la mà de Chris Watkins al desenvolupar el Q-learning. Aquest nou mètode estenia i integrava les tres branques, mencionades al principi, en la cerca de l'aprenentatge per reforç.

2.3. Elements de l'aprenentatge per reforç

En aquest apartat es vol definir la terminologia emprada en el projecte ja que és necessari per comprendre les explicacions que precedeixen el capítol.

Agent (agent). L'agent és l'objecte que aprèn. Per exemple, en un robot, la part responsable de realitzar les decisions s'anomena agent. Aquest sovint és l'ordinador o microcontrolador. Els braços d'un robot formarien part de l'entorn.

Entorn (enviroment). L'entorn o medi és tot allò que estigui fora de l'agent. Per exemple, en un laberint l'entorn el formaria les parets i la sortida.

Més enllà de l'agent i el medi, es poden definir quatre subelements importants d'un sistema d'aprenentatge per reforç: *política (policy)*, una *funció recompensa (reward function)*, una *funció valor (value function)* i opcionalment un *model* del medi.

Política (policy). Determina el comportament de l'agent. Representa la probabilitat d'escollir una acció determinada. Aquesta pot ser totalment estocàstiques, determinística o entre les dues (per exemple ε-greedy).

Funció recompensa (reward function). La recompensa és un valor numèric que portarà a l'agent cap al seu objectiu. Per tant, una funció recompensa defineix els events bons i dolents per l'agent.

Funció valor (value function). Una funció recompensa indica quant bo és l'estat immediat, mentre que, la funció valor és la suma de totes

les recompenses immediates indicant al final quines accions i estats s'han de prendre.

Model (model). És una representació matemàtica, computacional, etc. que permet reproduir el comportament d'un sistema. En aquest cas, les plantes que s'utilitzaran en l'estudi podrien representar sistemes físics com el d'un pèndul, un helicòpter, etc.

2.4. Aprenentatge per reforç

En aquest apartat es vol mostrar el problema de l'aprenentatge per reforç, és a dir, tot el que comporta un algoritme d'aquest tipus sense entrar en especificacions del mètode d'aprenentatge.

2.4.1. La interfície agent-medi

La problemàtica de l'aprenentatge per reforç va dirigida a aprendre de la interacció des d'un sistema per arribar a l'objectiu que es desitja. L'aprenent i el qui pren les decisions s'anomena agent i aquest interactua amb tot el que està al seu voltant que s'anomena entorn o medi. Aquesta interacció contínua produeix la selecció d'accions de l'agent i l'entorn li retorna les accions amb noves situacions de l'agent, és a dir, l'entorn retorna un estat nou.

L'entorn proporciona recompenses, aquestes són d'un valor el qual l'agent tractarà de maximitzar amb el pas del temps.

El conjunt actua en temps discret, ***t=0,1,2,3,..*** , llavors, a cada pas de temps, ***t***, l'agent i l'entorn interaccionen. En un pas de temps l'agent rep un estat, s_t, provinent de l'entorn el qual estarà dins del conjunt d'estats possibles, ***S***, que s'hagin determinat prèviament, $s_t \in S$. L'agent, partint de l'estat que proporciona l'entorn, ha de seleccionar una acció, a_t, la qual també estarà dins d'un conjunt d'accions possibles, $A(s_t)$, a realitzar, $a_t \in A(s_t)$. Un cop s'aplica l'acció, a_t, al següent pas de temps, ***t+1***, i com a conseqüència de l'acció anterior

s'obté una recompensa numèrica, r_{t+1}, la qual avalua l'acció anterior segons el nou estat que ha propiciat, s_{t+1}. La recompensa, així com els estats i les accions, correspondrà a un valor dins d'un conjunt predeterminat prèviament, $r_{t+1} \in R$.

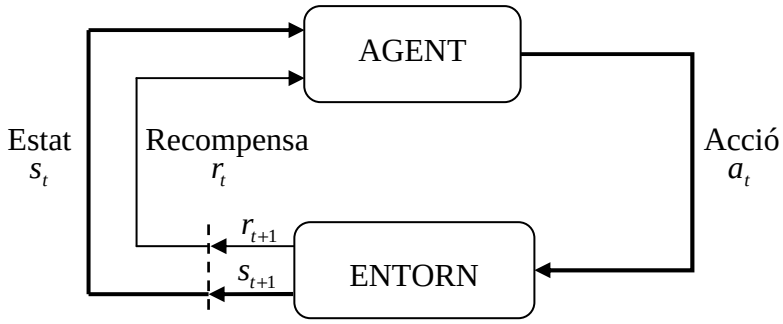


Figura 2.1. *Diagrama de la interacció agent-entorn*

L'agent inicialment crea un mapa on té en compte els conjunts d'estats, ***S***, i accions, ***A(s)***. A cada pas de temps es va registrant les possibilitats de seleccionar una acció o una altra. Aquest mapejat s'anomena política π_t, on $\pi_t(s,a)$ és la probabilitat que una acció sigui $a_t=a$ si l'estat és $s_t=s$. Els mètodes d'aprenentatge per reforç especifiquen com l'agent va canviant la seva política com a resultat de la seva experiència.

El funcionament general d'aquests tipus de mètodes és abstracte, no té cap indicació específica sobre el que pot o no treballar, permetent ser aplicat en diferents problemes de diverses maneres. Per exemple, el step de temps no cal que estigui referenciat a un interval de temps real, es pot referenciar a passos successius d'estat.

2.4.2. Objectius i recompenses

En l'aprenentatge per reforç, el propòsit o objectiu de l'agent queda reflectit sobre un senyal especial, anomenada recompensa, que passa de l'entorn a l'agent. La recompensa és tan sols un únic número el valor del qual varia d'un pas a l'altre. Dit d'altra manera, l'objectiu de

l'agent és maximitzar la quantitat total de recompensa obtingut. Això vol dir que no només s'ha de maximitzar la recompensa immediata sinó que també la recompensa acumulada.

La utilització d'un senyal de recompensa per formalitzar la idea d'un objectiu és una de les característiques més distintives de l'aprenentatge per reforç. Tot i que pot semblar que la manera de formular l'objectiu pugui estar limitat, a la pràctica està comprovat que és flexible i àmpliament aplicable a molts camps. Per exemple, per fer que un robot aprengui a caminar, l'investigador haurà previst una recompensa a cada pas de temps en que el robot avança. En funció de si aquest avança obtindrà una recompensa de +1 però si pel contrari retrocedís obtindria una recompensa de -1. D'aquesta manera es premiaria el fet d'avançar i aconseguir arribar a l'objectiu.

Tenint en compte l'exemple anterior s'observa que el senyal de recompensa no és el millor lloc per establir a l'agent el coneixement de com realment es vol arribar al que es desitja. Simplement se li marca l'objectiu principal. Per tant, es definirà l'objectiu de l'agent al medi, i per això cal definir bé quin és el medi de l'agent.

2.4.3. Retorns

Com es diu en l'apartat anterior l'agent busca maximitzar les recompenses, o també anomenat retorns, però no només a efectes immediats sinó a llarg termini, que és el que realment interessa. Una seqüència de recompenses rebudes al llarg del temps, tenint en compte el pas de temps ***t*** establert, s'anotaria com r_{t+1}, r_{t+2}, r_{t+3}**...**. Si el que es busca és maximitzar el retorn esperat, on el retorn és R_t, es defineix com una funció de la seqüència de recompensa. En un cas simple, aquest retorn serà la suma de totes les recompenses:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_T \qquad (2.1)$$

on ***T*** és el temps final. Aquesta aproximació pren sentit quan s'aplica en sistemes que tenen una noció natural de pas de temps i arriben a un temps final. Des del primer pas de temps fins aquest pas de temps

final es consideraria com a episodi. Cada episodi acaba en un estat especial anomenat *estat terminal* (terminal state), un cop acabat l'episodi es procedeix a reiniciar el sistema portant-lo a un estat inicial estàndard o predeterminat per l'usuari. Tasques amb episodis d'aquest tipus s'anomenen *tasques episòdiques*. En tasques episòdiques sovint cal distingir el conjunt d'estats no terminals, $\boldsymbol{S}$, del conjunt de tots els estats més l'estat terminal, $\boldsymbol{S^+}$.

Per altra banda, en molts casos la interacció agent-medi no té una forma episòdica, sinó que simplement avança en el temps sense límit. Per exemple, aquesta seria la formulació d'una tasca contínua de control sobre un procés, o el d'una aplicació que té una llarga durada. Aquestes s'anomenen *tasques contínues*. El plantejament del retorn en aquest tipus de tasques és problemàtic doncs el temps seria $\boldsymbol{T=\infty}$, i, per tant, seria absurd tractar de maximitzar un valor que tendeix a infinit ja que mai s'acabaria de formular el retorn.

El que es necessita en aquesta ocasió és el concepte de *descomptar* (discounting). L'agent tracta de seleccionar accions de manera que la suma de les recompenses que rep en el futur es maximitzi. En particular, l'agent partint de l'estat en que es troba l'entorn tria una acció $\boldsymbol{a_t}$ per maximitzar el benefici esperat de retorn, el *discounting return*:

$$R_t = r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \qquad (2.2)$$

On $\boldsymbol{\gamma}$ és un paràmetre, $\boldsymbol{0 \leq \gamma \leq 1}$, anomenat *proporció de descompte* (discount rate).

La proporció de descompte (discount rate, $\boldsymbol{\gamma}$) determina el valor present de les recompenses futures, és a dir, una recompensa que es rep en un instant de temps $\boldsymbol{t}$ té un alt valor al ser immediat però perd presència en el futur. Si $\boldsymbol{\gamma<1}$, la suma serà tan gran com la recompensa $\boldsymbol{\{r_k\}}$ estigui limitada. Si $\boldsymbol{\gamma=0}$, l'agent és "miop" en tant que només es preocupa per maximitzar les recompenses immediates: el seu objectiu en aquest cas és aprendre com escollir $\boldsymbol{a_t}$ per

maximitzar només el r_{t+1}. A mida que γ s'aproxima a 1, l'objectiu pren les recompenses futures molt més en compte: l'agent té més visió de futur.

2.4.4. La propietat de Markov

En el marc de treball de l'aprenentatge per reforç l'agent pren les decisions en base a un senyal provinent de l'entorn que s'anomena ***estat***. Del senyal d'estat no s'ha d'esperar que informi a l'agent de tot l'entorn en que es troba a cada instant de temps, o de tot allò que pugui ser útil per prendre decisions. Sempre hi haurà informació a la que no té accés ja sigui perquè està amagada o perquè no es pot transferir.

El que fora ideal seria un senyal estat que resumís i compactés informació passada que fos rellevant. Un senyal estat que sigui capaç de retenir tota la informació rellevant se sol dir que té *la propietat Markov*.

A continuació es defineix la propietat de Markov pel problema d'aprenentatge per reforç. Per fer-ho senzill, s'assumeix que hi ha un nombre finit d'estats i de valors de recompensa. Això permet parlar en termes de sumes i probabilitats en comptes d'integrals i densitats de probabilitat. Suposem un entorn general, com respondria aquest entorn al temps **t+1** a una acció feta al temps **t**? A mode de resum, la resposta estarà condicionada per tot el que ha succeït abans del temps **t+1**. En aquest cas la dinàmica només es pot definir mitjançant l'especificació completa de la distribució de probabilitat:

$$\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, ..., r_1, s_0, a_0 \mid\} \qquad (2.3)$$

per tot s', r, i tots els possibles valors d'events passats: s_t, a_t, r_t....., r_1, s_0, a_0. Si el senyal d'estat té la propietat Markov, llavors la resposta de l'entorn en $t+1$ depèn únicament de les representacions de l'estat i l'acció en l'instant t, en què la dinàmica de l'entorn es pot definir mitjançant l'especificació de només:

$$\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t \mid\} \quad (2.4)$$

per tot ***s'***, ***r***, i $\boldsymbol{a_t}$.

Dit d'una altra forma, si el temps ***t+1*** es considera com un temps futur, el temps ***t*** com el present i els temps ***t-1,...,0*** com el passat, llavors l'estat en un temps futur ***t+1*** només dependrà d'allò que succeeix al temps ***t***, és a dir, els estats futurs depenen del que passa en el present, ***t***, i no d'estats passats, ***t-1,...0***.

Si un entorn té la propietat de Markov, llavors el seu pas dinàmic (2.4) habilita per predir l'estat següent i esperar la següent recompensa assignada a l'estat i acció actual. S'observa que iterant aquesta equació es podrà predir tots els estats futurs i les recompenses partint del coneixement exclusiu de l'estat actual així com també seria possible tenint tot l'historial fins al moment actual. És la millor política per escollir les accions i per tant formar una funció pel senyal d'estat.

La propietat de Markov és important en l'aprenentatge per reforç, perquè s'assumeix que les decisions i els valors només són funció de l'estat actual.

2.4.5. Markov Decision Processes

Una tasca d'aprenentatge per reforç que satisfà la propietat de Markov s'anomena *Markov decision process*, o MDP. Si l'espai de l'estat i acció són finits, llavors s'anomena *finite Markov decision process* (finite MDP). Els MDP finits són importants en la teoria d'aprenentatge per reforç per l'ús que normalment es fa.

Un finite MDP es defineix pel seu conjunt d'estats i conjunt d'accions i per la dinàmica d'un sol pas de l'entorn. Donat un estat i acció, ***s*** i ***a***, la probabilitat de cada possible estat següent, ***s'***, és:

$$P^{a}_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\} \quad (2.5)$$

Molt similar, donat un estat qualsevol i una acció, ***s*** i ***a***, junts amb qualsevol estat següent, ***s'***, el valor esperat de la següent recompensa és:

$$R^{a}_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\} \quad (2.6)$$

Aquestes dues quantitats, $P^{a}_{ss'}$ i $R^{a}_{ss'}$, especifiquen completament els aspectes més importants de la dinàmica d'un finite MDP (només es perd informació sobre la recompensa que pot rebre del conjunt de recompenses possibles).

2.4.6. Funcions valor

Gairebé tots els algoritmes d'aprenentatge per reforç estan basats en funcions valor, funcions d'estats, o parell d'estat-acció, que estimen com de bo és per l'agent posar-lo en un nou estat. L'expressió "com de bo" es correspon amb la recompensa esperada. La recompensa que l'agent pot esperar rebre en un futur depèn de les accions que es prenen.

Recordant l'apartat 2.4.1, una política, ***π***, és un mapejat format per dos conjunts: el conjunt dels estats, ***s*** $\in$ ***S***, on es descriu tots els possibles estats del sistema, i el conjunt de les accions, ***a*** $\in$ ***A(s)***, on es descriuen totes les accions possibles a realitzar. La política ***π(s,a)*** descriu la probabilitat que hi ha de prendre una acció ***a*** quan s'està en un estat ***s***. Tenint en compte tot lo anterior, el valor corresponent a un estat sota una política, escrit com $V^{\pi}(s)$, és el retorn esperat partint de l'estat ***s*** i a continuació aplicar la política. Per MDPs es pot definir $V^{\pi}(s)$ com:

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \mid s_t = s\} \quad (2.7)$$

On $E_{\pi}\{\}$ denota el valor esperat, ja que l'agent segueix la política ***π***, i ***t*** és un pas de temps qualsevol.

De manera similar, es defineix el valor que es pren de l'acció ***a*** en l'estat ***s*** sota la política ***π***, ***$Q_\pi(s,a)$***, com el retorn esperat a partir de l'estat ***s***, prenent l'acció ***a***, i posteriorment seguint la política ***π***:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \mid s_t = s, a_t = a\} \qquad (2.8)$$

Una propietat fonamental de les funcions valor que s'utilitza al llarg de l'aprenentatge per reforç i la programació dinàmica és que es compleixin en particular les relacions recursives. Per una política qualsevol ***π*** i de qualsevol estat ***s***, la següent condició compleix que entre el valor de ***s***, i el valor dels seus estats successors possibles:

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\{R_t \mid s_t = s\} \\ &= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \mid s_t = s\} \\ &= E_{\pi}\left\{r_{t+1} + \gamma \cdot \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+2} \mid s_t = s\right\} \\ &= \sum_{a} \pi(s,a) \sum_{s'} P^a_{ss'} \cdot \left[R^a_{ss'} + \gamma \cdot E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+2} \mid s_{+1t} = s'\right\}\right] \\ &= \sum_{''} \pi(s,a) \sum_{s'} P^a_{ss'} \cdot \left[R^a_{ss'} + \gamma \cdot V^{\pi}(s')\right] \end{aligned} \qquad (2.9)$$

On és implícit que les accions, ***a***, s'agafen del conjunt ***A(s)***, i l'estat següent, ***s'***, s'agafen del conjunt ***S***, o del ***S^+*** en el cas d'un problema episòdic. L'última equació és *l'equació de Bellman per* ***V^π***. Expressa una relació entre el valor d'un estat i els valors dels seus estats següents.

2.4.7. Funcions valor òptimes

Resoldre una tasca d'aprenentatge per reforç significa trobar una política que aconsegueixi una gran quantitat de recompensa a llarg termini. Per les MDPs finites, es pot precisar una política òptima de la següent manera. Una política serà millor que una altra quan aquesta té un retorn per cada estat més alt que l'altre, dit d'una altra forma, ***$\pi \geq \pi'$*** si i només si ***$V^\pi(s) \geq V^{\pi'}(s)$*** per sot ***$s \in S$***. Hi ha moltes polítiques

per un sol sistema, o problema a tractar, però sempre hi ha una que és millor o similar a altres, i la millor política s'anomena *política òptima*. De polítiques òptimes com a mínim hi ha una, i es descriuen per $\boldsymbol{\pi^*}$. Aquestes comparteixen la mateixa funció valor, anomenada la funció valor òptima, $\boldsymbol{V^*}$, definida com:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{2.10}$$

per tot $\boldsymbol{s} \in \boldsymbol{S}$.

Les polítiques òptimes també comparteixen la mateixa funció òptima acció-valor, $\boldsymbol{Q^*}$, definida com:

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{2.11}$$

per tot $\boldsymbol{s} \in \boldsymbol{S}$ i $\boldsymbol{a} \in \boldsymbol{A(s)}$. Pel parell estat-acció (s,a), la funció dóna el retorn esperat per prendre l'acció $\boldsymbol{a}$ en un estat $\boldsymbol{s}$ després d'haver seguit una política òptima. Així, es pot escriure $\boldsymbol{Q^*}$ en termes de $\boldsymbol{V^*}$:

$$Q^*(s,a) = E\{r_{t+1} + \gamma \cdot V^*(s_{t+1}) \mid s_t = s, a_t = a\} \tag{2.12}$$

CAPÍTOL 3: ESTUDI DE TÈCNIQUES D'APRENENTATGE PER REFORÇ

En aquest capítol es tracta d'analitzar i entendre, de manera general, els diferents mètodes existents en l'aprenentatge per reforç, que es poden englobar dins de tres grans grups: la Programació Dinàmica, els Mètodes de Monte Carlo i l'Aprenentatge en diferència temporal.

Aquests tres grups contenen diferents algoritmes, i estan separats d'aquesta manera degut a la seva manera de tractar el problema. A més, cada grup té propietats avantatjoses que els altres no tenen així com també tenen inconvenients, com pot ser la complexitat de traduir la matemàtica.

El contingut del capítol és un resum extret del llibre escrit per Sutton i Barto[1], si es desitja conèixer en més detall els algoritmes que es presenten i d'altres que no es fa referència en l'estudi, cal adreçar-se als capítols 4, 5 i 6 del llibre esmentat.

3.1. Programació dinàmica

Les tècniques de Programació Dinàmica (PD) es refereixen a aquells algoritmes que poden ser emprats per obtenir polítiques òptimes donat el coneixement complet d'un model perfecte de l'entorn com un Procés de Decisió de Markov. Així com la resta de tècniques d'aprenentatge per reforç, la programació dinàmica es basa en el manteniment de les funcions de valor de cada estat, V(s) i/o Q(s,a), per estructurar el coneixement. Obtenint els valors òptims d'ambdues funcions s'obté una representació òptima de la política a seguir. Aquestes funcions de valor òptimes compleixen les equacions d'optimalitat de Bellman (equacions 3.1 i 3.2), i que formulen matemàticament el principi d'optimalitat, que es descriu a continuació:

Principi d'Optimalitat. *Una política òptima té la propietat que qualsevol que sigui l'estat inicial i la primera decisió presa; la resta de decisions que es prenen han de formar una política òptima respecte l'estat resultant de la primera decisió.*

$$V^*(s) = \max_a E\left\{r_{t+1} + \gamma \cdot V^*(s_{t+1}) \mid s_t = s, a_t = a\right\} = \\ = \max_a \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma \cdot V^*(s')\right] \tag{3.1}$$

$$Q^*(s,a) = E\{r_{t+1} + \gamma \cdot \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a\} = \\ = \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma \cdot \max_{a'} Q^*(s', a')\right] \tag{3.2}$$

per tot $\boldsymbol{s} \in \boldsymbol{S}$, $\boldsymbol{a} \in \boldsymbol{A(s)}$ i $\boldsymbol{s'} \in \boldsymbol{S}$.

3.1.1. Iteració de política

La majoria de les tècniques de programació dinàmica es deriven de les equacions (3.1) i (3.2). L'algoritme d'Iteració de Política (Policy Iteration) consisteix en una repetició iterativa de dos passos fonamentals. Per una banda, l'avaluació de la política de la qual es disposa en un determinat moment, és a dir, el càlcul de la seva funció

de valor-estat. Per l'altra banda, la millora de la política mitjançant aquesta funció.

A continuació es mostra l'algoritme en pseudocodi en la figura següent.

Iteració de la Política

- Inicialització
  $V(s) \in \mathbb{R}$ i $\pi(s) \in A(s)$ arbitraris per tot $s \in S$.
- Repetir
  1. Avaluació de la política
     - Repetir
       - $\Delta \leftarrow 0$
       - Per cada $s \in S$
         - $v \leftarrow V(s)$
         - $V(s) \leftarrow \sum_{s'} P_{ss'}^{\pi(s)} \left[R_{ss'}^{\pi(s)} + \gamma V(s') \right]$
         - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
     - Fins que $\Delta < \theta$ (un número positiu enter)
  2. Millora de Política
     - política_estable $\leftarrow$ cert
     - Per cada $s \in S$
       - $b \leftarrow \pi(s)$
       - $\pi(s) \leftarrow \arg\max_a \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V(s') \right]$
       - Si $b \neq \pi(s)$, llavors política_estable $\leftarrow$ fals

Fins que política_estable = cert

Figura 3.1. *Algoritme pseudocodi Iteració de Política*

3.1.2. Iteració del valor

La iteració del valor requereix un nombre infinit d'iteracions per convergir exactament a V*. A la pràctica, la iteració de la funció valor para quan les diferències que hi ha són molt petites.

Per evitar tenir que recalcular la funció valor de la política a cada iteració de l'algoritme, com al cas anterior, s'utilitza aquest algoritme (Value Iteration), que es mostra en la figura 3.2.

Iteració del Valor

- Inicialitzar V(s) arbitràriament. Per exemple, V(s)=0, $\forall s \in S$
- Repetir
  - $\Delta \leftarrow 0$
  - Per tot $s \in S$
    - $v \leftarrow V(s)$
    - $V(s) \leftarrow \max_a \sum_{s'} P_{ss'}^{\pi(s)} \left[R_{ss'}^{\pi(s)} + \gamma \cdot V(s') \right]$
    - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
- Fins que $\Delta < \theta$ (un número positiu enter)
- Donar com a sortida una política π tal que $\pi(s) = \arg\max_a \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma \cdot V(s') \right]$

Figura 3.2. *Algoritme pseudocodi Iteració del Valor*

La iteració del valor combina, a cada repetició, una escombrada d'avaluació de la política i una escombrada de la millora de la política. Per convergir de manera més ràpida se sol afegir diverses escombrades de l'avaluació de la política entre l'escombrada de la millora de la política.

3.1.3. Programació dinàmica asíncrona

Un dels principals inconvenients que presenten els algoritmes de programació dinàmica fins ara tractats és que totes les operacions involucrades s'han de realitzar sobre tot el conjunt d'estats. D'aquesta manera, per sistemes amb un conjunt d'estats molt gran el problema podria fer-se intractable.

Els algoritmes de programació dinàmica asíncrona estan estructurats de manera que només se selecciona l'estat que realment és necessari i útil per l'aprenentatge evitant així haver d'escombrar tot el conjunt d'estats. Per tant, de tot el conjunt d'estats hi haurà una part que rebrà més actualitzacions que no pas la resta, producte de la importància que tenen uns sobre els altres. A més a més, aquests mètodes permeten ser utilitzats mentre s'interactua amb un entorn real, és a dir, l'algoritme de programació dinàmica permet ser executat mentre s'està experimentant amb el MDP. L'experiència de l'agent pot ser emprada per determinar sobre quins estats del MDP realitzar actualitzacions. A la vegada, la informació actual de la política i les funcions valor poden determinar quines accions pot realitzar l'agent en el seu aprenentatge, podent prendre més atenció a parts de l'entorn que es consideren més rellevants.

3.2. Mètodes de Monte Carlo

A diferència del mètode anterior, programació dinàmica, per aquest no necessita tenir un coneixement de l'entorn tan profund ja que no té la necessitat de conèixer les seves dinàmiques. Els mètodes de Monte Carlo es basen en dades, és a dir, només cal proporcionar a l'algoritme seqüències dels estats, de les accions i de les recompenses rebudes d'un entorn real o simulat. Aprendre del conjunt de dades *reals* no requereix cap coneixement previ de la dinàmica de l'entorn i, tot i així, es pot aconseguir un comportament òptim. Per altra banda, aprendre de dades *simulades* també permet obtenir resultats potents.

Tot i que es necessita un model, aquest només necessita generar transicions de la mostra, no les distribucions de probabilitat completa de totes les possibles transicions que calen pels mètodes de programació dinàmica.

Els mètodes de Monte Carlo són formes de resoldre el problema d'aprenentatge per reforç basat en les mitjanes de retorn de la mostra. Es definirà els mètodes de Monte Carlo com a tasques episòdiques. Aquest mètode és molt similar al de la programació dinàmica.

Aquests mètodes es basen en l'actualització de les funcions valor mitjançant seqüències o intents complets de resolució del problema per part de l'agent en una interacció directa amb l'entorn, és a dir, que els resultats s'obtenen després de la interacció de l'episodi per resoldre el problema en comptes de fer-ho per pas de temps. Donat que en principi el model del problema no és conegut, és necessari aprendre la funció de valor-acció en lloc de la funció de valor-estat, ja que s'ha d'estimar explícitament el valor de cada acció amb la finalitat de suggerir una política. Per tant, l'objectiu dels mètodes Monte Carlo és estimar Q*.

3.2.1. Estimació Monte Carlo dels valors d'acció

Si no es disposa d'un model llavors és més útil estimar els valors de les accions que no pas el valor dels estats. Amb un model, amb només el valor dels estats és suficient per determinar una política; simplement es mirarà per escollir la millor combinació de retorn i pròxim estat. Quan no es té un model els estats no són suficients, per aquesta raó cal estimar les accions per saber quines d'aquestes són útils i poder així suggerir una política. Per tant, un dels objectius importants pels mètodes Monte Carlo és estimar Q*.

L'única complicació és que un grup important de parelles estat-acció mai es visitaran. Si π és una política determinística, llavors en el seguiment de la política π s'observarà retorns per una de les accions de cada estat. Si no hi han retorns als que fer la mitja llavors les estimacions de Monte Carlo no milloren perque es desitjaria poder avaluar tot el conjunt d'accions possibles que té cada estat, plantejant així un problema seriós per l'algoritme.

Aquest és el problema general de mantenir l'exploració (maintaining exploration). Per a que l'avaluació de la política funcioni amb els valors d'acció, cal assegurar una exploració contínua. Una manera de fer-ho és especificant que al primer pas de cada episodi comenci en una parella estat-acció, i totes les parelles tindran una probabilitat diferent

de zero de ser seleccionades com a inici d'un episodi. Això garanteix que tots els parells estat-acció es visitin un nombre infinit de vegades en el límit d'un nombre infinit d'episodis. Això s'anomena *exploracions d'inici* (exploring starts).

3.2.2. Control Monte Carlo

A continuació es considera utilitzar Monte Carlo com a control. La idea general és seguir el que s'ha explicat en programació dinàmica, aproximar a l'òptim. Per una banda la funció valor es va iterant per aproximar-se a l'òptim en la política d'aquell moment, i per l'altra banda, la política també s'itera per millorar-la.

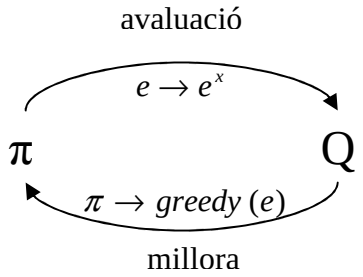


Figura 3.3. *Funcionament general de Monte Carlo*

Aquests dos canvis treballen un contra l'altra fins a cert punt, ja que cada un fa millores respecte l'altre, el símil seria el gos que es mossega la cua. Però en conjunt provoquen que la política i la funció valor s'aproximin a l'òptim.

Es considerarà una versió de Monte Carlo clàssica de la iteració de política. Per a aquest mètode s'actua alternant dos passos: en primer lloc es fa l'avaluació de política i en segon lloc es procedeix a la millora de la política. Començant amb una política qualsevol $\boldsymbol{\pi_0}$, aplicant el procediment anterior dels dos passos s'avança arribant al final a una política òptima i a la funció acció-valor òptima.

$$\pi_0 \overset{E}{\rightarrow} Q^{\pi_0} \overset{I}{\rightarrow} \pi_1 \overset{E}{\rightarrow} Q^{\pi_1} \overset{I}{\rightarrow} \pi_2 \overset{E}{\rightarrow} \cdots \overset{I}{\rightarrow} \pi^* \overset{E}{\rightarrow} Q^*$$

on $\overset{E}{\rightarrow}$ significa que es realitza una avaluació completa de la política i $\overset{I}{\rightarrow}$ significa que es realitza una millora completa de la política.

Per a l'avaluació de la política de Monte Carlo és natural alternar entre avaluació i millora de forma episodi per episodi. Després de cada episodi, el retorn obtingut s'utilitza per l'avaluació de la política, i llavors la política és millorada en tots els estats que s'han visitat en l'episodi. S'anomena algoritme Monte Carlo ES (Monte Carlo with Exploring Starts), i a continuació es mostra el codi.

Monte Carlo ES

- Inicialitzar, per tot $s \in S$, $a \in A(s)$:
  - $Q(s,a) \leftarrow$ valor arbitrari
  - $\pi(s) \leftarrow$ valor arbitrari
  - Retorns(s,a) ← llista buida
- Repetir per sempre
  1. Generar un episodi utilitzant arrancada exploratoria i π
  2. Per cada parell (s,a) que aparèix a l'episodi
     - R ← retorn o guany obtingut després de la primera aparició del parell (s,a)
     - Afegir R a Retorns(s,a)
     - $Q(s,a) \leftarrow$ promig(Retorns(s,a))
  3. Per cada s en l'episodi
     $\pi(s) \leftarrow \arg\max_a Q(s,a)$

Figura 3.4. *Algoritme pseudocodi Monte Carlo ES*

En aquest algoritme s'utilitza la llista de Retorns(s,a), també anomenada guanys, per emmagatzemar la llista de retorns que s'han anat obtenint al executar l'acció **a** des de l'estat **s**. Posteriorment, s'utilitza el valor mig d'aquesta llista per actualitzar la funció Q. La llista de retorns és important en els mètodes Monte Carlo, però també

hi ha mètodes derivats de l'anterior que no necessiten d'aquesta llista ja que van calculant les mitjanes sense haver d'emmagatzemar-les en una llista.. Com es pot observar, aquest algoritme manté els dos passos fonamentals d'avaluació de la política: l'actualització de la funció Q i de millora de la política; i l'actualització de π.

L'algoritme, figura 3.4, mostra com a l'inici sempre s'arrenca de manera que la primera parella estat-acció se selecciona aleatòriament, el qual assegura que tots els parells estats-acció són visitats infinites vegades, assegurant així la convergència de l'algoritme.

3.3. Temporal difference learning

L'aprenentatge per diferència temporal (temporal difference (TD) learning) és una combinació de les idees de Monte Carlo i les idees de la programació dinàmica (DP). Els mètodes TD, com els mètodes Monte Carlo, poden aprendre directament de l'experiència sense un model de la dinàmica de l'entorn. Els mètodes TD, com els DP, realitzen estimacions en funció d'altres estimacions prèviament apreses.

3.3.1. Predicció de la TD

Els mètodes de Monte Carlo i TD utilitzen l'experiència, conjunt de dades d'estat, accions i retorns, per resoldre el problema de la predicció. Donada una experiència i aplicant una política π, ambdós mètodes actualitzen la seva estimació **V** de $\mathbf{V^{\pi}}$. Si un estat $\mathbf{s_t}$ és visitat a temps **t**, llavors els dos mètodes actualitzen l'estimació $\mathbf{V(s_t)}$ basant-se en el que succeeix després de la visita.

Un mètode de Monte Carlo apte per entorns no estacionaris és

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)] \tag{3.3}$$

on $\mathbf{R_t}$ és el retorn després d'un temps **t** i **α** és un paràmetre constant de la mida del pas. Aquest mètode s'anomena *constant – α MC*. Mentre els mètodes de Monte Carlo han d'esperar fins arribar al final de cada episodi per determinar l'actualització en $\mathbf{V(s_t)}$, els mètodes TD necessiten esperar només fins al pròxim pas de temps. Un TD, al temps **t+1** realitza una actualització utilitzant la recompensa $\mathbf{r_{t+1}}$ i l'estimació $\mathbf{V(s_{t+1})}$ observats. El mètode més simple de TD es coneix com TD(0).

Algoritme TD(0)

- Inicialitzar V(s) arbitràriament
- Repetir (per cada episodi)
  - a ← acció donada per π i s
  - Executar l'acció a i observar el reforç rebut, r, i l'estat següent, s'
  - $V(s) \leftarrow V(s) + \alpha[r + \gamma \cdot V(s') - V(s)]$
  - s ← s'

Fins que s sigui un estat terminal

Figura 3.5. *Algoritme pseudocodi TD(0) per estimar* V^{π}

- Avantatges

Els mètodes TD tenen un avantatge important sobre els mètodes DP, no és necessari tenir un model de l'entorn que li proporcioni tots els estats possibles ni la recompensa per cadascun d'aquests estats. Tot i així, també pot funcionar amb un model.

L'avantatge sobre els mètodes Monte Carlo és que amb els mètodes Monte Carlo per conèixer els resultats s'ha d'esperar fins al final d'un episodi perque és quan fan l'actualització. Mentre, amb els mètodes TD només cal esperar un pas de temps, per tant es pot saber l'evolució de l'aprenentatge de manera més detallada. A més a més, els mètodes TD convergeixen més ràpidament que els mètodes Monte Carlo.

3.3.2. Sarsa: amb política de control

Hi ha dos tipus de classes, els que tenen una política i els que no en tenen. En aquest cas es tracta Sarsa amb política. Per aquest mètode és preferible aprendre una funció acció-valor que no pas una funció estat-valor, a més, tenint política cal estimar la funció $Q^{\pi}(s,a)$ per la política que té i pel conjunt d'estats i accions. Per aprendre i estimar la funció Q^{π} s'utilitza la mateixa metodologia que abans s'ha descrit per aprendre V^{π} pel TD. Seguint la metodologia esmentada i afegint que es tracta de parelles estat-acció el funcionament seria com el que es descriu en el diagrama següent:

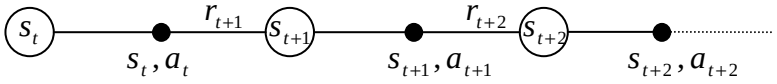


Figura 3.6. *Diagrama del funcionament amb Sarsa*

En l'anterior apartat es considerava les transicions només fetes d'estat a estat i s'aprenia el valor dels estats. Ara, amb el Sarsa és té en compte una parella i per tant, es consideren les transicions com parella d'estat-acció a parella d'estat-acció. En el fons són exactament el mateix, les dues opcions són cadenes de Markov amb un procés de recompensa. El teorema per convergir la funció estat-valor utilitzat pel mètode TD(0) pot ser aplicat en aquest cas, modificant les pertinents parts, deixant l'expressió de l'algoritme Sarsa com es mostra:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \qquad (3.4)$$

L'algoritme es va actualitzant a cada transició mentre l'estat s_t no sigui terminal. Si l'estat s_{t+1} fos terminal llavors l'estimació $Q(s_{t+1}, a_{t+1})$ per aquest estat es definiria com zero. Aquesta norma utilitza tots els elements de la tupla dels events, $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$, que constitueixen una transició d'una parella estat-acció a la següent. Aquesta tupla de cinc dóna nom a l'algoritme de *Sarsa*.

És senzill dissenyar un algoritme de control amb política basada en el mètode de predicció Sarsa. Com en tots els mètodes amb política,

contínuament s'estima $\mathbf{Q^{\pi}}$ pel comportament de la política **π**, i a la vegada canvia π respecte $\mathbf{Q^{\pi}}$. La forma general de l'algoritme de control Sarsa es mostra en la figura 3.7.

Algoritme Sarsa

- Inicialitzar Q(s,a) arbitràriament
- Repetir (per cada episodi)
  - Inicialitzar s
  - Triar a des d'una s utilitzant la política derivada de Q (per exemple, e-greedy)
  - Repetri (per cada pas de episodi):
    - Prendre acció a, observar r, s'
    - Triar a' des d'un s' utilitzant la política derivada de Q (per exemple, e-greedy)

      $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \cdot Q(s',a') - Q(s,a)\right]$

      $s \leftarrow s', a \leftarrow a'$

  Fins que s sigui un estat terminal

Figura 3.7. *Algoritme pseudocodi Sarsa amb política*

Les propietats de convergència de l'algoritme Sarsa depèn de la dependència de la política sobre Q. Per exemple es pot utilitzar polítiques $\in$ -greedy o $\in$ -soft.

3.3.3. Q-learning: control TD sense política

Un dels avenços més importants en l'aprenentatge per reforç va ser el desenvolupament d'un algoritme de control TD sense política conegut com Q-Learning (Watkins, 1989). La seva forma més senzilla, un pas Q-learning es defineix com:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \qquad (3.5)$$

En aquest cas, la funció acció-valor apresa, **Q**, aproxima directament a **Q***, la funció òptima acció-valor, independentment de la política seguida. Això simplifica l'anàlisi de l'algoritme i habilita la convergència de manera ràpida. Tot i no tenir una política per definició, conté un mecanisme que determina quin parell d'estat-acció s'ha de visitar i actualitzar. Tanmateix, cal que tots els parells contínuament siguin actualitzats per tal que convergeixi correctament. El procediment de l'algoritme Q-learning es mostra a continuació, figura 3.8.

Algoritme Q-learning

- Inicialitzar Q(s,a) arbitràriament
- Repetir (per cada episodi)
  - Inicialitzar s
  - Repetri (per cada pas de episodi):
    - Seleccionar una acció a a partir de s utilitzant una política derivada de Q
    - Executar l'acció a observant l'esforç immediat rebut, r, i elsegüent estat, s'
    - Actualitzar l'entrada de la taula, Q(s,a) amb l'equació:
      $$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)\right]$$
    - $s \leftarrow s'$

  Fins que s sigui un estat final

Figura 3.8. *Algoritme pseudocodi Q-learning*

3.4. Selecció de la tècnica

Els sistemes que s'estudiaran en aquest projecte no defineixen l'entorn d'un sistema real, per tant, s'assumirà que els sistemes no són una representació completa amb coneixement profund d'aquest. Llavors, els mètodes de programació dinàmica quedarien al marge d'aquest estudi per aquest motiu. S'ha de tenir en compte que els sistemes mai

arriben a ser perfectes i per tant, aplicar un mètode de programació dinàmica seria complicat.

Els possibles mètodes a aplicar serien els Monte Carlo o els Temporal Difference. Ambdós s'ajusten al que s'ha comentat anteriorment; necessiten un model per funcionar, però no cal que defineixin el sistema amb perfecció, doncs es basen en l'experiència i l'estimació. Tots dos són bons però els mètodes TD s'adeqüen millor doncs fan un tractament més fi, és a dir, mentre els mètodes Monte Carlo cal esperar un episodi per obtenir una millora de les funcions, els mètodes TD reuneix les característiques de Monte Carlo i Programació Dinàmica realitzant millores a cada pas de temps obtenint millores de les funcions de forma més constant i ràpida. Permet veure l'evolució i el sistema "discret" al que estaria sotmès encaixa millor amb els models disponibles per realitzar l'estudi.

Dels mètodes TD s'han presentat alguns en l'apartat anterior, tant als llibres com a la xarxa s'hi poden trobar més mètodes i versions dels que s'han presentat en aquest estudi. D'entre els presentats en l'apartat anterior, Q-learning i Sarsa són molt semblants, però el mètode Sarsa té més paràmetres dels quals depèn. Tenint en compte que els models per aquest estudi són variats, és millor disposar de la possibilitat de modificar i/o ajustar varis paràmetres per afinar millor els controls a realitzar a aquests models. Per tant, s'utilitzarà el mètode Sarsa.

CAPÍTOL 4: DESCRIPCIÓ DELS SISTEMES

Per realitzar l'estudi sobre les possibilitats que té un algoritme d'aprenentatge per reforç es presenta a continuació un conjunt de models a tractar extrets del Benchmark 2008[5]. Un recull de models per realitzar un concurs promogut pels professors de la universitat de Terrassa ETSEIAT, del departament d'automàtica.

D'aquests models proposats s'escolliran un grup per aplicar-li l'esmentat algoritme. Tots ells són diferents entre si, per tant, representen un repte cada cas.

En aquest apartat es presentaran tots els models d'aquest benchmark i es realitzarà el control PID del conjunt escollit per més endavant, observar les diferències amb el mètode Sarsa.

4.1. Descripció del Benchmark

Amb motiu del IFAC Workshop on Digital Control Past, Present and Future of PID Control que es va portar a terme a Terrassa el més d'Abril del 2000[5], els professors Astrom i Hagglund van presentar un benchmark compost per diversos sistemes per avaluar el disseny dels controladors PID que van tenir un fort impacte durant l'esmentat Workshop.

Com es pot observar es tracta de sistemes monovariables en general lineals, excepte el sisè sistema, que cobreixen àmpliament totes les possibles dinàmiques que es poden trobar en processos reals.

El benchmark proposat té les següents precisions:

Sistema amb múltiples pols: n=8

Sistema amb quatre pols amb el següent paràmetre: α=0.5

Sistema amb un zero positiu: α=5

Sistema de primer ordre amb retard pur: T=0.1 s

Sistema de segon ordre amb retard pur: T=0.1 s

Sistema de conducció tèrmica (no entrava al concurs)

Sistema amb modes ràpids i lents igual al proposat per Åström i Hägglund

Sistema marginalment estable igual al proposat per Åström i Hägglund

Sistema oscil·latori: ζ=0.1 i ω_0=5 rad/s

Sistema inestable igual al proposat per Åström i Hägglund

Sistema amb acció integral (no entrava al concurs)

Els controladors PID proposats podien tenir una estructura estàndard (accions P, I i D independents i en paral·lel a la sortida de l'error de regulació) o estructura en dos graus de llibertat (PI-D o I-PD) amb o sense filtre de l'acció derivativa i amb o sense "anti-windup".

Cadascun dels nou controladors que es proposaven serien avaluats en simulació (SIMULINK) amb el sistema corresponent. Se'ls aplicaria una consigna (c) en forma d'esglaó d'amplitud inferior o igual a la unitat i una variació de càrrega (p) en forma de pols també d'amplitud igual o inferior a la unitat amb una duració superior al minut. La variable manipulada (u) estaria limitada a 10 vegades el seu valor en règim permanent com resposta a l'esglaó de consigna sense pertorbacions. No es tenia en compte el soroll de mesura ni soroll en el procés.

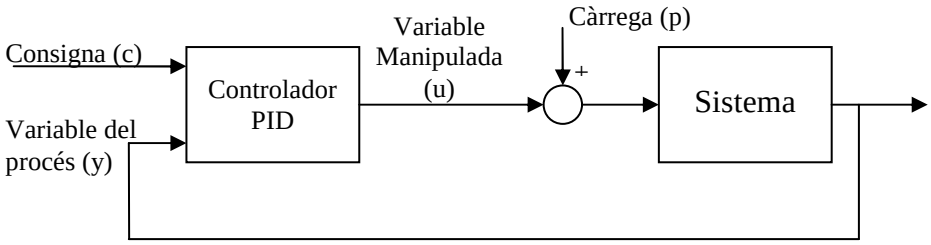


Figura 4.1. *Esquema general del sistema controlat*

4.2. Característiques dels sistemes

En aquest apartat hi ha una petita descripció de cada sistema proposat, extret del Benchmark 2008 on fa referència a l'original realitzat per K. J. Åström i T. Hägglund.

4.2.1. Sistema amb múltiples pols

Funció de transferència:

$$G(s) = \frac{1}{(s+1)^n} \qquad \text{n = 1,2,3,4,8} \tag{4.1}$$

Aquests sistemes són molt comuns. Per n = 1 i 2 es pot aconseguir controlar per PI o PID respectivament. Per valors més alts de n el sistema es comporta com sistemes amb llargs temps morts. Aquests sistemes s'han utilitzat per fabricants de controladors, com els casos

de prova durant molt de temps. Informació obtinguda d'Eurotherm i Foxboro.

4.2.2. Sistema amb quatre pols

Funció de transferència:

$$G(s) = \frac{1}{(s+1)(1+\alpha \cdot s)(1+\alpha^2 \cdot s)(1+\alpha^3 \cdot s)} \qquad \alpha = 0.1,0.2,0.5,1.0 \tag{4.2}$$

Aquest sistema té quatre pols els quals la seva separació ve determinada pel paràmetre α. Per valors petits hi ha millores dràstiques quan es passa d'un control PI a PID. Per α =1 el sistema és igual al sistema (4.1) per n=4.

4.2.3. Sistema amb un zero positiu

Funció de transferència:

$$G(s) = \frac{1-\alpha \cdot s}{(s+1)^3} \qquad \alpha = 0.1,0.2,0.5,1,2,5 \tag{4.3}$$

Aquest sistema té tres pols iguals i un zero al semiplà dret. El rendiment que es pot assolir ve determinat pel paràmetre α. La dificultat del control augmenta amb l'augment de α.

4.2.4. Sistema de primer ordre amb retard pur

Funció de transferència:

$$G(s) = \frac{1}{1+sT} e^{-s} \qquad T = 0,0.1,0.2,0.5,2,5,10 \tag{4.4}$$

Aquest és el sistema clàssic que s'ha utilitzat en moltes investigacions de control PID. El sistema redueix a un retard pur mitjançant T = 0 i representa el sistema dominat pel retard en funció de T. Moltes de les primeres regles d'ajust es van basar en aquest model. Un

desavantatge d'aquest model és que, tenint l'estructura d'un passa baixos, té una caiguda, de la magnitud (dB), baixa a freqüències altes.

4.2.5. Sistema de segon ordre amb retard pur

Funció de transferència:

$$G(s) = \frac{1}{(1+sT)^2} e^{-s} \quad \text{T} = 0,0.1,0.2,0.5,2,5,10 \tag{4.5}$$

El sistema és similar a l'anterior però té més caiguda, de la magnitud (dB), a freqüències altes. El sistema redueix a un retard pur per T = 0.

4.2.6. Sistema de conducció tèrmica

Funció de transferència:

$$G(s) = e^{-\sqrt{s}} \tag{4.6}$$

El sistema representa les dinàmiques d'un sistema de conducció tèrmica d'una dimensió. El guany definitiu és $k_u = e^{\pi}$. Implementacions analògiques d'aquest sistema s'han utilitzat a Eurotherm per testejar controladors de temperatura.

4.2.7. Sistema amb modes ràpids i lents

Funció de transferència:

$$G(s) = \frac{100}{(s+10)^2}\left(\frac{1}{s+1} + \frac{0.5}{s+0.05}\right) \tag{4.7}$$

La dinàmica essencial d'aquest sistema té una ràpida constant de temps T = 1 amb un guany moderat (1) i una lenta constat de temps T = 20 amb un guany alt (10). Regles d'ajust simple basats en la resposta esglaó normalment no dóna un bon ajust per aquests sistemes d'aquest tipus perquè és difícil obtenir una bona estimació del guany i de la constant de temps.

4.2.8. Sistema marginalment estable

Funció de transferència:

$$G(s) = \frac{(s+6)^2}{s(s+1)^2(s+36)} \tag{4.8}$$

El sistema és marginalment estable.

4.2.9. Sistema oscil·latori

Funció de transferència:

$$G(s) = \frac{\omega_0^2}{(s+1)(s^2+2\zeta\omega_0 s+\omega_0^2)} \quad \zeta = 0.1, \quad \omega_0=1,2,5,10 \tag{4.9}$$

Sistemes d'aquest tipus amb poc amortiment, ζ, no són bons candidats per un control PID. El sistema és fàcil de controlar si ω_0 és gran. El comportament es pot millorar dràsticament mitjançant més estructures de control.

4.2.10. Sistema inestable

Funció de transferència:

$$G(s) = \frac{1}{(s^2-1)} \tag{4.10}$$

Aquest és un model simple d'un pèndul invertit.

4.2.11. Sistema amb acció integral

És molt útil tenir sistemes amb acció integral. Una bona col·lecció es pot aconseguir afegint un integrador als sistemes 1-5.

4.3. Selecció dels sistemes

En aquest apartat es mostra la selecció de sistemes que es tractaran amb l'algoritme d'aprenentatge per reforç. A més, es mostra el control clàssic, tipus PID, aplicat per controlar-los a tal efecte de poder realitzar comparatives més endavant amb el sistema de control per reforç.

Per cada model, escollit per l'estudi, s'ha calculat un controlador a fi de realitzar una comparativa amb el resultat de l'algoritme aplicat. Hi ha diversos mètodes per sintonitzar PIDs: analítics, empírics, prova-error, per optimització, etc. El desenvolupament de mètodes de sintonització és extens des que Ziegler i Nichols van proposar el seu procediment al 1942.

Pels casos a estudiar s'opta per un mètode empíric el qual, normalment, es divideix en mètodes de llaç obert i mètodes de llaç tancat. Els mètodes empírics permeten sintonitzar un PID exigint, sovint, unes condicions del comportament com ara, el temps emprat per portar el sistema a la consigna desitjada, l'esmorteïment, etc. Tot i que per trobar els controls PID cal seguir aquestes condicions, no és l'objectiu principal de l'estudi sintonitzar uns controladors que busquin errors zero, per tant, els controls que es calculen a continuació no seran els òptims però compliran amb la seva funció i s'ajustaran per obtenir errors petits i en poc temps.

El mètode utilitzat per sintonitzar els PIDs en els models seleccionats per realitzar l'estudi és el conegut Ziegler i Nichols, per la seva facilitat i afinitat per la seva utilització en altres projectes. No obstant, es poden utilitzar altres tècniques del mateix estil com Cohen i Coon, Kaya i Sheib, Lopez Murril i Smith, etc.

4.3.1. Cas 1

Es tracta del sistema amb múltiples pols (apartat 4.2.1), amb n = 8.

$$G(s) = \frac{1}{(s+1)^8} = \frac{1}{s^8 + 8s^7 + 28s^6 + 56s^5 + 70s^4 + 56s^3 + 28s^2 + 8s^1 + 1} \tag{4.11}$$

Per obtenir el controlador PID utilitzant el mètode Ziegler-Nichols hi ha dues possibilitats, com es comenta anteriorment, aplicant el mètode de llaç obert o aplicant el mètode de llaç tancat. Ambdues requereixen del compliment de propietats per poder-les aplicar amb la seguretat d'obtenir un controlador que funcioni.

Aplicar el mètode del llaç obert implica la següent propietat:

$$0.1 \leq T_0 / T_p \leq 1.0 \tag{4.12}$$

On T_0 i T_p són paràmetres que es calculen a partir de l'obtenció de dades del llaç obert.

En aquest cas, amb aquesta base (4.11), la propietat (4.12) no compleix i per tant s'aplicarà el llaç tancat que sí compleix. (Per saber més consultar **[4]**).

Per trobar el PID en llaç tancat cal trobar primer uns paràmetres necessaris per saber finalment els valors de Kp, Ki i Kd. Els resultats obtinguts de buscar la Kpcrítica i el Tosc (temps d'oscil·lació en Kpcrítica) són:

$$\mathrm{k_{pcrítica}} = 1.88 \qquad \mathrm{T_{OSC}} = 15.5\mathrm{s}$$

Utilitzant les taules de Ziegler i Nichols, que es poden trobar al llibre de Control Modern de Ogata, s'obtenen els següents valors:

$$T_I = T_{OSC} / 2 = \frac{15.5}{2} = 7.75$$
$$T_D = T_{OSC} / 8 = \frac{15.5}{8} = 1.937$$

$$K_P = 0.6 \cdot K_{PCrític} = 0.6 \cdot 1.88 = 1.128$$
$$K_I = \frac{K_P}{T_I} = \frac{1.128}{7.75} = 0.1455$$
$$K_D = K_P \cdot T_D = 1.128 \cdot 1.937 = 2.185$$

Els resultats d'aplicar aquest control PID es mostren en les següents gràfiques:

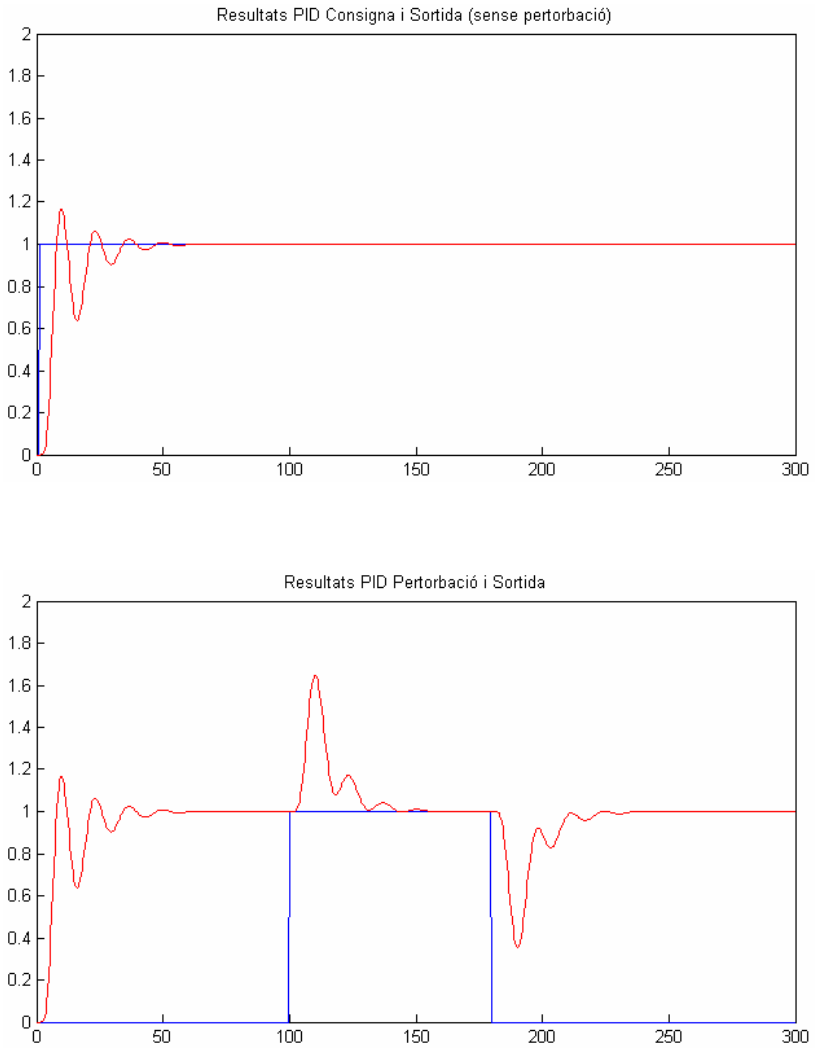


Figura 4.2. *Resultats del sistema amb el PID, cas 1*

Com s'observa, no és un control molt ràpid. Es podria dir que han de passar uns 50 segons aproximadament per tal que el sistema estigui controlat. Pel que fa a la pertorbació afegida, el control també tarda uns 50 segons en estabilitzar el senyal. (En tot moment, el senyal vermell representa la sortida i la blava representa la consigna o pertorbació).

No és el millor PID que se li pot aplicar, però per l'estudi que es realitzarà més endavant, aplicant un algoritme d'aprenentatge per reforç és més que suficient.

4.3.2. Cas 2

Es tracta del sistema amb un zero positiu (apartat 4.2.3), fent que es tracti un sistema que és inestable. Pel cas es tractarà amb una α=5, és a dir, es tractarà amb un grau de dificultat alt respecte les altres possibilitats.

$$G(s) = \frac{1+\alpha \cdot s}{(s+1)^3} = \frac{1+5 \cdot s}{(s+1)^3} \qquad (4.13)$$

El procediment per obtenir el PID per aquest sistema segueix el mateix mètode que en el cas anterior. El llaç obert, utilitzant Ziegler-Nichols, ha de complir la propietat (4.12) però la prova feta a la planta (4.13) per comprovar-ho revela que no compleix i per tant, no s'obtindria un bon controlador. Per aquesta raó s'aplica el mètode de llaç tancat, i els dos paràmetres necessaris per calcular el controlador, realitzant la prova pertinent són:

$$k_{\text{pcrítica}} = 0.5008 \qquad T_{OSC} = 9\text{s}$$

Utilitzant les taules de Ziegler i Nichols s'obté el següent:

$$T_I = T_{OSC} / 2 = \frac{9}{2} = 4.5$$

$$T_D = T_{OSC} / 8 = \frac{9}{8} = 1.125$$

$$K_P = 0.6 \cdot K_{PCrític} = 0.6 \cdot 0.5008 = 0.30048$$

$$K_I = \frac{K_P}{T_I} = \frac{0.30048}{4.5} = 0.06677\widehat{3}$$

$$K_D = K_P \cdot T_D = 0.30048 \cdot 1.125 = 0.33804$$

Els resultats que s'obtenen amb aquest controlador són els mostrats per les següents gràfiques.

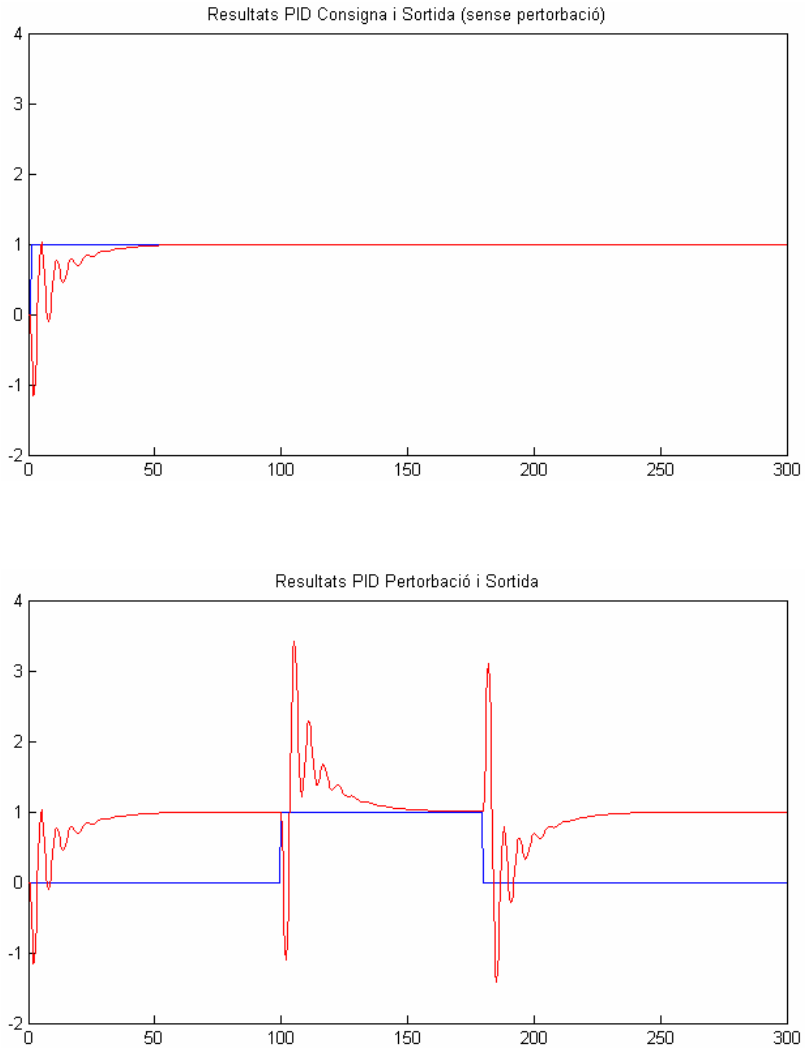


Figura 4.3. *Resultats del sistema amb el PID, cas 2*

El control quan s'aplica a un sistema sense pertorbacions es podria dir que actua prou bé, però quan es troba amb una pertorbació, aquesta afecta molt al senyal resultant. És per això que s'ha tractat de trobar uns paràmetres del PID per millorar la resposta. Aquesta cerca de paràmetres parteix dels ja trobats i, simplement partint de l'observació, es modifiquen alguns valors per tractar de millorar el comportament en front pertorbacions.

$$K_P = 0.1$$
$$K_I = 0.067$$
$$K_D = 0.33804$$

S'ha vist que reduint l'acció proporcional del controlador, l'error produït en el senyal resultant és sensiblement menor, i per això s'ha optat per establir aquest com el controlador del sistema.

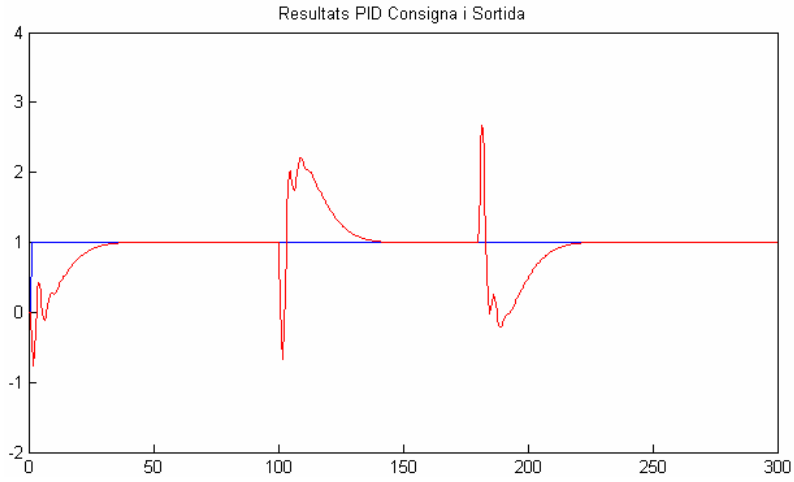


Figura 4.4. *Resultat amb el nou control, cas 2*

En la figura anterior si es compara amb les anteriors gràfiques, es comprova que el resultat millora, l'error en excés que es produeix no és tan alt i s'estalvia part de les oscil·lacions. No es tracta de trobar el millor PID pel sistema, doncs no és l'objectiu del projecte, però es pretén obtenir un model per ser comparat posteriorment amb l'algoritme Sarsa.

4.3.3. Cas 3

El sistema escollit per aquest tercer cas és el sistema de segon ordre amb retard pur (apartat 4.2.5), amb un T=0.1.

El programa Simulink de Matlab disposa de blocs i funcions que permeten introduir un retard a un sistema, però s'ha optat per realitzar una simplificació del retard pur utilitzant una tècnica. La raó d'aquesta simplificació és que, a l'hora de realitzar l'aprenentatge per reforç, la utilització d'una eina de Matlab per introduir el retard impedeix el seu aprenentatge. Els estats interns del model es mantenen a zero si el pas de temps aplicat a la simulació és menor a la durada del retard bloquejant l'aprenentatge. Per això s'ha preferit aplicar la simplificació i mantenir la igualtat de condicions en els dos sistemes de control estudiats.

Per aproximar la funció de transferència del retard $e^{-\theta s}$ per un sistema de dimensió finita existeixen varies tècniques.

En aquest cas s'utilitzarà una aproximació de Pade d'ordre *q* en el numerador i d'ordre *p* al denominador, per fer-ho més fàcil es tractarà com si fos p=q (ordre relatiu 0).

$$e^{-\theta s} = \frac{N_p}{D_p} \tag{4.14}$$

on:

$$N_p = \sum_{j=0}^{p} \frac{(2\cdot p - j)!\, p!}{(2\cdot p)!\, j!(p-j)!}(-\theta\cdot s)^j \qquad \mathrm{D}_p = \sum_{j=0}^{p} \frac{(2\cdot p - j)!\, p!}{(2\cdot p)!\, j!(p-j)!}(\theta\cdot s)^j \tag{4.15}$$

L'ordre de la simplificació del retard per la qual s'ha optat és de segon ordre, és a dir, p=2. Aplicar un primer ordre crea major distorsió que no pas un de segon ordre, i aplicar un ordre més alt la millora que es pot apreciar és poca. Per tant, la simplificació quedaria de la següent manera:

$$N_p = \sum_{j=0}^{p} \frac{(2\cdot p - j)!\, p!}{(2\cdot p)!\, j!(p-j)!}(-\theta\cdot s)^j = \frac{(4-0)!2!}{4!0!(2-0)!}(-s)^0 + \frac{(4-1)!2!}{4!1!(2-1)!}(-s)^1 + \frac{(4-2)!2!}{4!2!(2-2)!}(-s)^2 =$$
$$= \frac{48}{48} - \frac{12}{24}s + \frac{4}{48}s^2$$

$$\mathrm{D}_p = \sum_{j=0}^{p} \frac{(2\cdot p - j)!\, p!}{(2\cdot p)!\, j!(p-j)!}(\theta\cdot s)^j = \frac{(4-0)!2!}{4!0!(2-0)!}(s)^0 + \frac{(4-1)!2!}{4!1!(2-1)!}(s)^1 + \frac{(4-2)!2!}{4!2!(2-2)!}(s)^2 =$$
$$= \frac{48}{48} + \frac{12}{24}s + \frac{4}{48}s^2$$

$$e^{-s} = \frac{\frac{48}{48} - \frac{12}{24}s + \frac{4}{48}s^2}{\frac{48}{48} + \frac{12}{24}s + \frac{4}{48}s^2} = \frac{1 - \frac{1}{2}s + \frac{1}{12}s^2}{1 + \frac{1}{2}s + \frac{1}{12}s^2} = \frac{s^2 - 6s + 12}{s^2 + 6s + 12} \qquad (4.16)$$

Utilitzant l'expressió (4.16) substituint-la a l'expressió del sistema (4.5), s'obté el nou sistema a analitzar (4.17).

$$G(s) = \frac{1}{(1+0.1\cdot s)^2} e^{-s} = \frac{1}{(1+0.1\cdot s)^2} \cdot \frac{(s^2 - 6s + 12)}{(s^2 - 6s + 12)}$$

$$G(s) = \frac{s^2 - 6s + 12}{0.01s^4 + 0.26s^3 + 2.32s^2 + 8.4s + 12} \qquad (4.17)$$

Un cop ja es té el sistema final, per trobar els paràmetres del controlador PID s'ha utilitzat l'eina de Matlab, Sisotool. S'ha utilitzat aquesta eina ja que realitzar-ho a mà era complicat, doncs no s'acabava de trobar els paràmetres. Aquesta eina disposa del autotunning, i s'ha seleccionat el PID.

El format que es mostra en la finestra de l'eina com a resultat de la cerca del PID, per Ziegler i Nichols, és:

$$C = 0.56226 \frac{(1+0.17s)(1+0.98s)}{s} \qquad (4.18)$$

Per obtenir els paràmetres Kp, Ki i Kd cal que el format sigui el següent:

$$C = \frac{Kd \cdot s^2 + Ki \cdot s + Kp}{s} \tag{4.19}$$

Per tant, els paràmetres de control PID del sistema són:

$$K_P = 0.6466$$
$$K_I = 0.5623$$
$$K_D = 0.09367$$

En les següents gràfiques es mostra el resultat obtingut amb aquest PID.

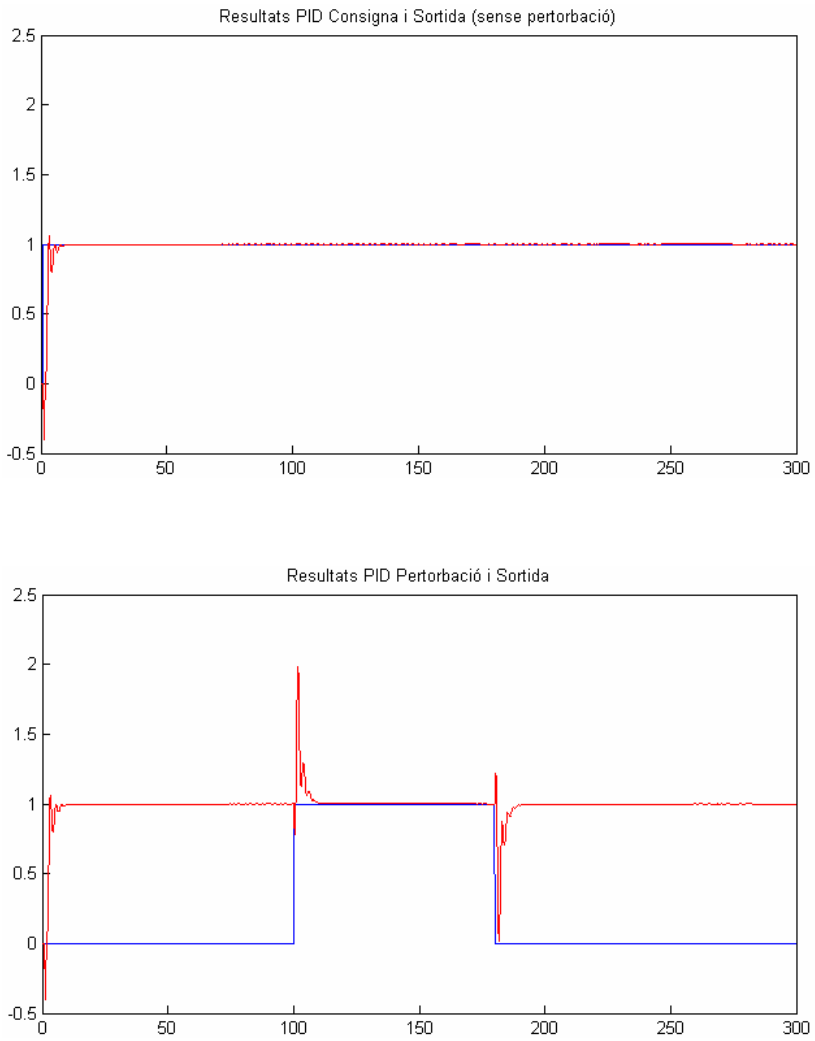


Figura 4.5. *Resultats del sistema amb el PID, cas 3*

En la figura 4.5 es veuen les gràfiques on es pot concloure que el control actua segons el que s'espera, complint les especificacions que comporta un control Ziegler-Nichols, tot i no ser l'objectiu principal. Actua amb rapidesa, de manera que el temps que roman inestable és poc, i la seva diferència respecte la consigna és de les més petites, si es comparessin amb altres simplificacions del retard.

4.3.4. Cas 4

Per aquest últim cas s'ha triat el sistema oscil·latori (apartat 4.2.9), amb els paràmetres següents: $\zeta=0.1$ i $\omega_0=5$.

$$G(s)=\frac{\omega_0^2}{(s+1)\left(s^2+2\zeta\omega_0 s+\omega_0^2\right)}=\frac{25}{s^3+2\cdot s^2+26\cdot s+25} \qquad (4.20)$$

L'obtenció del regulador PID s'ha efectuat mitjançant la tècnica de llaç obert doncs aquest sistema, amb aquesta configuració, permetia fer-ho. Els paràmetres obtinguts de l'execució del sistema en llaç obert són:

$$\mathrm{T}_1=\mathrm{T}_{28'3\%}=0.5$$
$$\mathrm{T}_2=\mathrm{T}_{63'2\%}=0.75$$
$$K=\frac{Y}{R}=\frac{1}{1}=1$$

Abans d'aplicar alguna tècnica s'han de revisar i comprovar que les expressions següents compleixen, d'aquesta manera es confirmarà si és possible realitzar un control partint de llaç obert.

$$\mathrm{T}_\mathrm{p}=1.5\cdot(T_2-T_1)=1.5\cdot(0.75-0.5)=0.375$$
$$\mathrm{T}_\mathrm{o}=\mathrm{T}_2-T_p=0.75-0.375=0.375$$

$$0.1\le\frac{\mathrm{T}_\mathrm{o}}{\mathrm{T}_\mathrm{p}}\le 1.0 \qquad (4.21)$$

$$0.1\le\frac{0.375}{0.375}\le 1.0$$

Com es veu, l'expressió (4.21) compleix i per tant es pot aplicar el mètode de llaç obert.

Per calcular els paràmetres del PID cal aplicar les següents fórmules de sintonia:

$$M_p = K \cdot K_p$$
$$M_i = \frac{T_p}{T_i}$$
$$M_d = \frac{T_d}{T_p}$$

(4.22)

On les M_p, M_i, M_d s'obtenen a partir de l'equació:

$$M_{p,i,d} = a \cdot \left(\frac{T_o}{T_p} \right)^b + c \qquad (4.23)$$

Si apliquem Ziegler i Nichols per un controlador PID amb c=0, les constants necessàries per realitzar els càlculs són:

Tipus	***a***	***b***
P	1.2	-1.0
I	0.5	-1.0
D	0.5	1.0

Utilitzant els paràmetres de la taula, el conjunt (4.22) i finalment l'expressió (4.23) s'obtindrà el controlador.

$$M_p = 1.2 \cdot \left(\frac{0.375}{0.375} \right)^{-1} = 1.2$$

$$M_i = 0.5 \cdot \left(\frac{0.375}{0.375} \right)^{-1} = 0.5$$

$$M_d = 0.5 \cdot \left(\frac{0.375}{0.375} \right)^{1} = 0.5$$

$$T_i = \frac{T_p}{M_i} = \frac{0.375}{0.5} = 0.75$$

$$T_d = M_p \cdot T_p = 0.5 \cdot 0.375 = 0.1875$$

$$K_p = \frac{M_p}{K} = \frac{1.2}{1} = 1.2$$

$$K_i = \frac{K_p}{T_i} = \frac{1.2}{0.75} = 1.6$$

$$K_d = K_p \cdot T_d = 1.2 \cdot 0.1875 = 0.225$$

El control que s'ha obtingut realitza la seva funció però no actua com es desitjaria, per aquesta raó s'ha ajustat el controlador mitjançant el prova i error fins obtenir un controlador amb un comportament que porta la base a la consigna desitjada amb més eficàcia. Els paràmetres finals pel PID són:

$$K_P = 0.5143$$

$$K_I = 0.4898$$

$$K_D = 0.135$$

Els resultats es poden veure reflectits en les següents gràfiques.

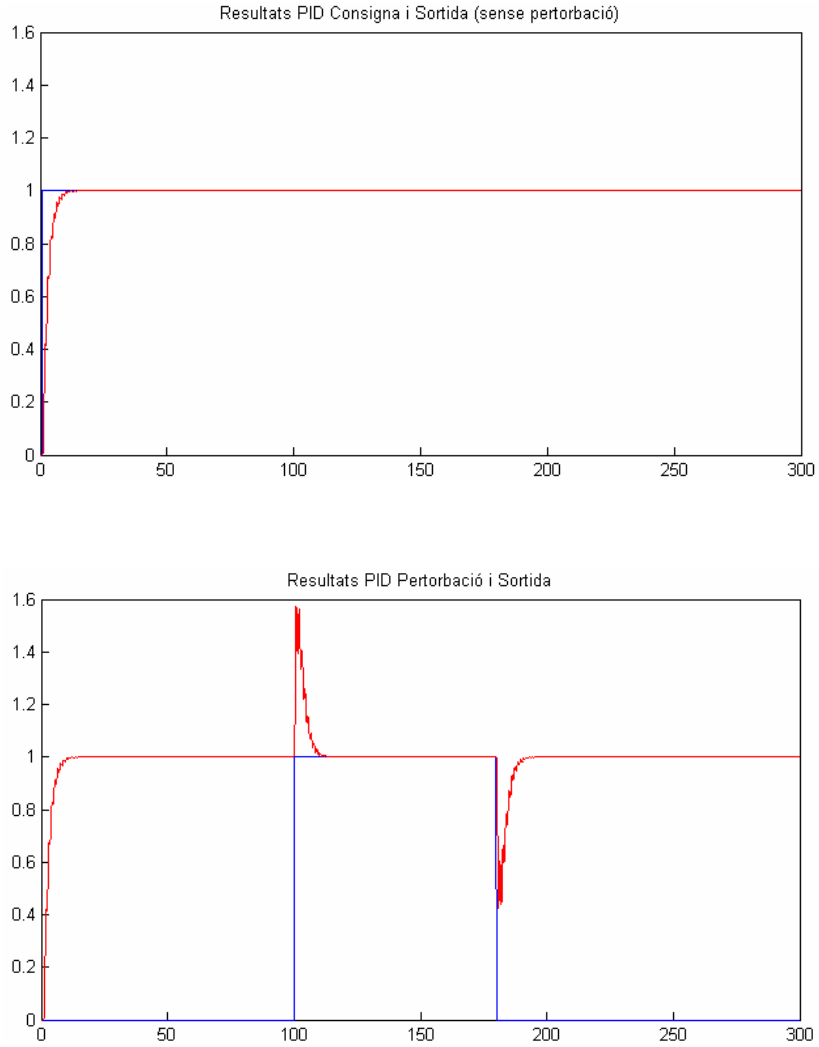


Figura 4.6. *Resultats del sistema amb el PID, cas 4*

Com es pot apreciar, el control actua ràpidament i les pertorbacions que pugui tenir el sistema aquest les elimina de forma efectiva, a més a més, compleix amb la relació d'esmorteïment segons Ziegler-Nichols.

CAPÍTOL 5: APLICACIÓ DE LA TÈCNICA SARSA

Com es comenta als apartats d'objectius i abast, es pretén controlar uns sistemes mitjançant una tècnica per reforç, i aquesta és Sarsa.

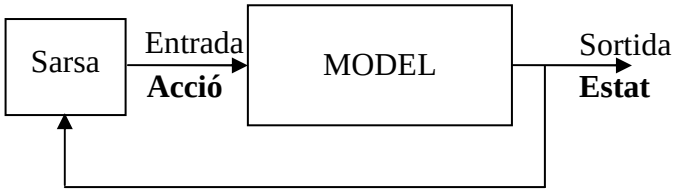
Per aplicar el mètode s'ha de tenir en compte certs aspectes, com ara l'estructura pel control o els paràmetres d'aprenentatge, que determinaran el funcionament de l'algoritme i conseqüentment el resultat. En aquest capítol s'explica quins punts són els més rellevants i les possibilitats estudiades per a cadascun d'ells, per finalment poder llegir els resultats amb la informació suficient i entendre com s'ha arribat a ells.

5.1. Estructura del control

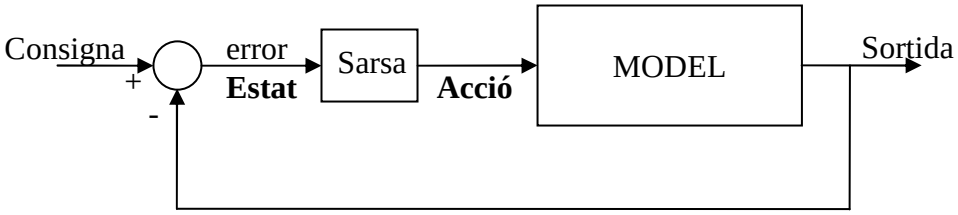
A l'hora de controlar un sistema es pot fer de diverses maneres: aplicar un filtre (tot i que és per sistemes molt senzill i, per tant, no és el més emprat), fer un control en llaç obert o fer el control en llaç

tancat. Es podria entrar a debatre si hi han més maneres de controlar un model en funció del nombre d'entrades i sortides, si és estable o inestable, etc. En aquest cas, només es tracten els casos de control més generals i representatius.

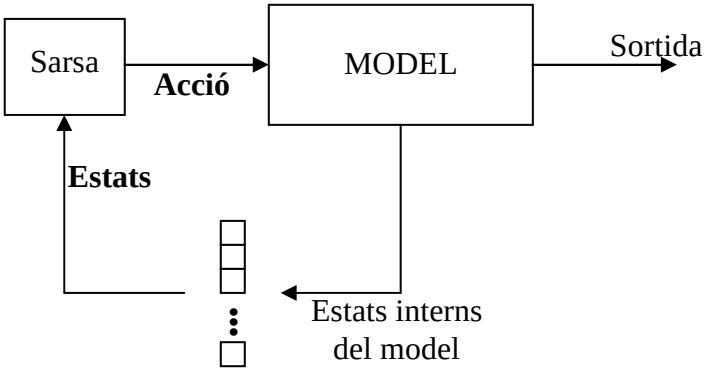
Les possibles estructures de control estudiades, o com a mínim s'han tingut en compte, es mostren a continuació.



a) Control simple



b) Control en llaç tancat



c) Control emprant els estats interns del model

Figura 5.1. *Diferents estructures de control*

La figura 5.1 mostra les diferents estructures presents en l'estudi, si més no, estan descrites com l'evolució que ha patit l'estructura de control de l'algoritme durant l'estudi.

L'opció A és la primera estructura que es va proposar per la simplicitat que presenta, és més fàcil de dominar i no mostra dificultats. Els ***estats*** són els valors de sortida del model i l'***acció*** es correspon amb l'acció de control que donaria un control PID al seu model, per tant, l'acció que proporciona l'algoritme dependrà en gran mesura de la sortida que s'obté i lo molt proper que estigui aquesta de l'objectiu o consigna.

Durant la fase de construcció de l'algoritme i part de la fase de proves va ser molt útil pel que s'ha comentat abans, però si s'analitza bé, no es pot deixar com a estructura de control. El perquè és el següent, no és exactament una estructura de control tal i com es coneix avui dia. Dit d'una altra manera, l'estructura té una realimentació però no s'utilitza per obtenir un error, com al control B, sinó per que l'algoritme pugui puntuar l'acció anterior i decidir quina és la millor acció en el següent pas de temps. No s'utilitza per transformar-lo sinó com a dada que necessita l'algoritme pel seu funcionament.

L'opció B de la figura 5.1 s'aproxima molt més al que es coneix com a control, on l'única diferència és que no hi ha un PID sinó el Sarsa. Els ***estats*** són els errors produïts de la sortida respecte la consigna marcada, i les ***accions*** són igual que a l'opció A. Aquesta estructura s'ajusta més en aquest estudi perquè es tracta de fer una comparativa amb el control clàssic PID, a l'hora de fer la comparativa ambdues parts estaran en igualtat de condicions. És per això que l'estructura de control B s'utilitza en l'estudi, no obstant, derivada d'aquesta estructura s'entra en l'última de les opcions mostrades.

L'estructura C, que utilitza els estats interns del model, és producte de cercar una millora en els resultats obtinguts en proves. En les anteriors estructures l'estat és un únic valor, per tant, es pot dir que la informació que s'obté és relativament pobre. El model d'un sistema, al passar-lo a espai d'estats, conté més informació que es podria tractar.

El problema és que suposaria, en una aplicació real, que el model fos conegut i no molt complex. Els sistemes que es tracten en aquest estudi no són models de processos reals, tot i ser emprats en proves com es comenta en apartats anteriors. El nombre d'estats interns dificultaria molt l'aprenentatge, doncs el nombre de parelles estat-acció vindrien determinades pel nombre d'estats i els seus possibles valors.

A més, es podria bifurcar en dues branques més aquesta estructura i entrar en la utilització d'aquesta amb control simple o en llaç tancat.

L'estructura clàssica de PID en llaç tancat és la més adient per l'estudi, però no es descartaria una possible ampliació en un futur amb un conjunt d'estats amb més informació, el qual podria millorar els resultats.

5.2. Estructura de l'algoritme

En aquest capítol es presenta l'estructura de tot l'algoritme, ja que està separat en dues parts per fer una millor diferenciació. Per una banda es presenta l'algoritme que realitza l'aprenentatge i per l'altra banda es troba l'algoritme de simulació.

S'intenta donar una idea del seu funcionament general i s'entra en algunes parts de l'algoritme, més en detall, per denotar la importància d'aquestes i les seves implicacions. En apartats posteriors es tracten aspectes d'alta importància amb més profunditat.

5.2.1. Estructura general de l'algoritme d'aprenentatge

Per construir l'estructura general dels casos que es tracten en aquest projecte es pren com a base l'exemple de l'exercici del cotxe i la muntanya, proposat al llibre *"Reinforcement Learning An Introduction" de Richard S. Sutton i Andrew G. Barto*[1], i realitzat per Jose Antonio Martin H.[10].

En la figura següent es mostra l'estructura principal per tal que el sistema pugui aprendre.

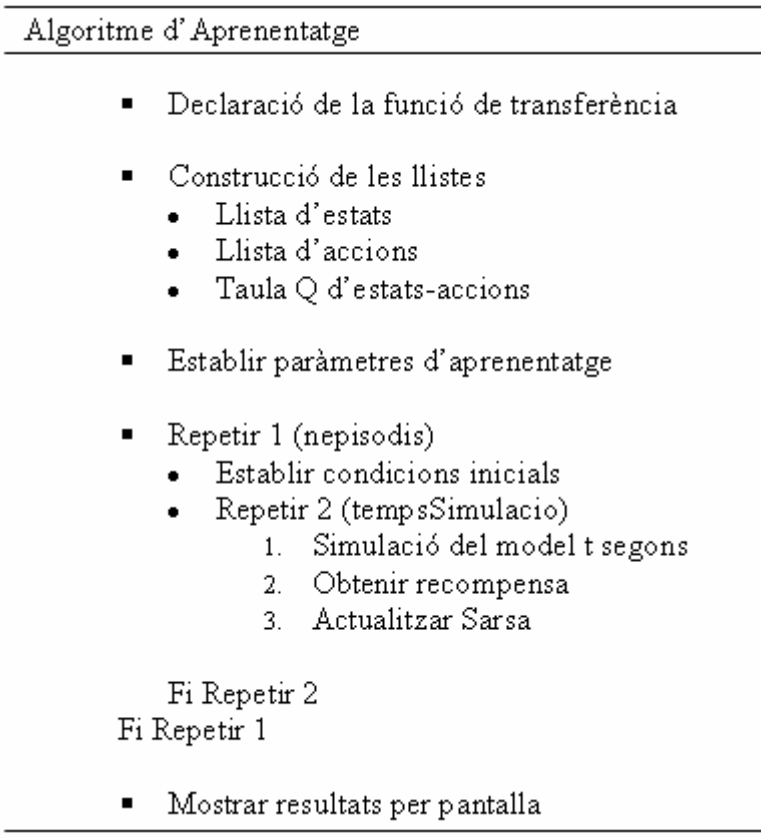


Figura 5.2. *Algoritme d'aprenentatge*

En primer lloc es declara la funció de transferència que defineix el sistema a tractar, aquest es passa a format d'espai d'estats per poder utilitzar l'eina de Matlab "lsim" la qual permet simular sistemes retornant els estats interns.

A continuació es construeixen les llistes dels estats i accions possibles. La definició dels estats i accions en part es fa per intuïció i basant-se en l'estudi anterior fet amb els controladors lineals. Per tenir una referència a l'hora de posar el rang de les accions s'observarà el senyal que surt del controlador PID, mentre que, per saber el rang dels estats només cal observar els valors de sortida del model.

En la següent figura, a tall d'exemple s'estudiaria el rang de les accions i el rang dels estats pel cas 1.

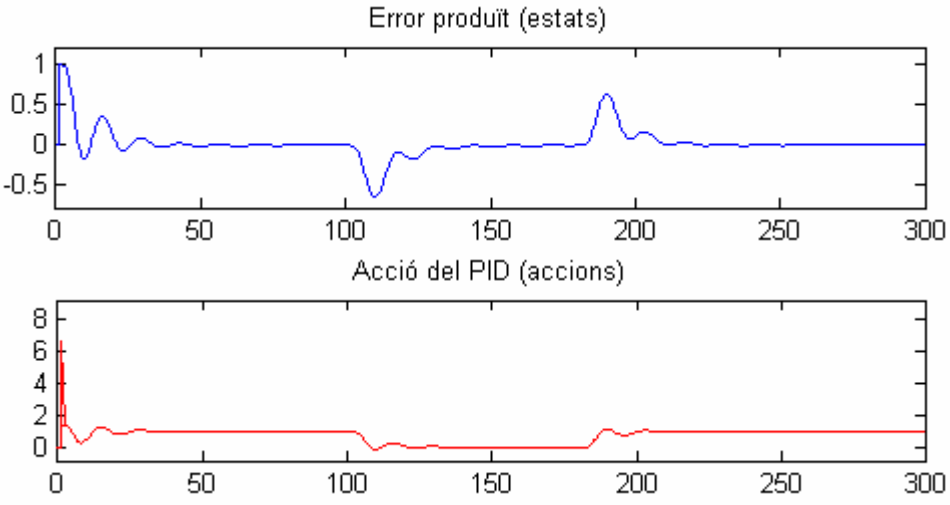


Figura 5.3. *Rang d'estats i accions, cas 1*

Pel cas 1, el rang dels estats quedaria emmarcat entre -1 i 1 mentre que les accions tenen un rang major perque les accions inicials són molt altes, quedaria entre -1 i 7.

Obtenir uns bons resultats dependrà en part d'aquestes llistes doncs donarà més o menys precisió. Les funcions que permeten construir els estats i accions són:

ConstLlistaEstats()

ConstLlistaAccions()

Un cop es tenen els estats i accions, es procedeix a construir la taula Q en la qual s'anirà construint i definint quins són les millors accions a aplicar en funció de l'estat del sistema assignant un valor, que anomenarem recompensa.

ConstTaulaQ(nestats,naccions)

Construïda la taula, es procedeix a establir valors als paràmetres que necessita l'algoritme Sarsa. Per saber més sobre la decisió del valor de cada paràmetre de l'algoritme, en apartats posteriors es tracta amb profunditat.

Entre els paràmetres de Sarsa es troba el nombre d'episodis que es desitja realitzar. Un episodi equival a la simulació del sistema durant un determinat temps (el temps en les simulacions és de 30 segons, però pot variar si escau) durant el qual l'algoritme va aprenent. Com més episodis es demanin més temps durarà l'aprenentatge, però s'ha de tenir en compte el temps de processat i si realment necessita realitzar tants episodis.

A cada episodi nou, s'inicialitzen les condicions inicials del model. Inicialment es va optar per fixar les condicions inicials de tal manera que es reforçaria sempre el mateix camí. Al començar l'episodi es trobaria amb:

- Estats interns del model a zero.
- Sortida zero.
- Error igual a la consigna, és a dir, 1.

La definició de les condicions inicials feta és prou lògica, si el que es desitja és arribar a un error zero, iniciant la simulació per aprendre sempre des del mateix punt afavoriria a l'aprenentatge reforçant més o menys el mateix camí, dins d'un marge. Això seria molt similar al que es pot trobar a la indústria, una màquina o procés pot començar la seva feina sempre d'un estat en concret ja sigui pel propi procés que així ho demana o per comoditat s'estableix. Però també és cert que hi ha processos que es paren i es reprenen obligant al controlador a realitzar de nou el control i, per tant, no es pot conèixer des de quin punt es parteix, llevat que es tingui el procés o màquina amb un equip de sensors i actuadors que identifiquin l'estat.

Com les condicions inicials no sempre es coneixen es va decidir donar un toc aleatori als estats interns del model.

- Estats interns del model aleatoris.
- Sortida, resultat dels estats interns aleatoris.
- Error, resultat de la diferència entre la consigna i la sortida.

No és que hi hagi una millora molt notable respecte unes condicions fixes però d'aquesta manera s'avarca molt més camp, al cap i a la fi, si

l'aprenentatge és bo acabarà reforçant un camí dins d'un marge, però amb la seguretat que s'exploren diferents possibilitats des del principi de la simulació.

En establir paràmetres, condicions inicials i nombre d'episodis es procedeix a realitzar la simulació del sistema aplicant l'algoritme Sarsa. Aquesta simulació es realitza dins un bucle el qual a cada iteració avançarà un temps "t" fins a realitzar una simulació total d'uns 10 segons com a mínim, segons el model a analitzar el temps pot ser augmentat. En aquest temps l'algoritme haurà aprés part del camí, per això, gràcies al nombre d'episodis designat abans, el sistema es tornarà a simular fins fer el total d'episodis. Cadascun d'aquests realitzats dóna l'oportunitat a l'algoritme a aprendre millor el camí per arribar a l'objectiu desitjat, que és obtenir una sortida de valor 1 constant amb error zero.

Per determinar el temps "t" que avança en la simulació se segueix el següent criteri. Segons Nyquist per preservar la informació original d'una senyal analògica al ser discretitzada en temps, la taxa de mostreig havia de ser major que el doble de la màxima component de freqüència de la senyal analògica. Llavors, per tal de preservar la informació del que s'obté al simular, el criteri a seguir és realitzar una prova senzilla al model. Aquesta consisteix en aplicar un graó unitari en llaç obert al model, en quan la resposta assoleix aproximadament el 63%, es prendrà el temps que ha tardat en arribar a aquest valor. El temps que tarda servirà per determinar la freqüència de mostreig, com ha de ser major que el doble de la màxima component de freqüència del senyal, se seguirà el criteri de dividir el temps entre 10.

5.2.2. Estructura de l'algoritme de simulació

Després de realitzar l'aprenentatge sobre el sistema a tractar i obtenir una taula que defineix el comportament a seguir, cal aplicar aquest comportament aprés al sistema i observar el resultat que s'obté. A

més és necessari comprovar si el que s'ha aprés serà capaç d'absorbir una pertorbació.

Per portar a terme tot el que s'ha esmentat es disposa d'un algoritme que permet efectuar tot el que s'ha explicat. A continuació es mostra l'esquema que segueix.

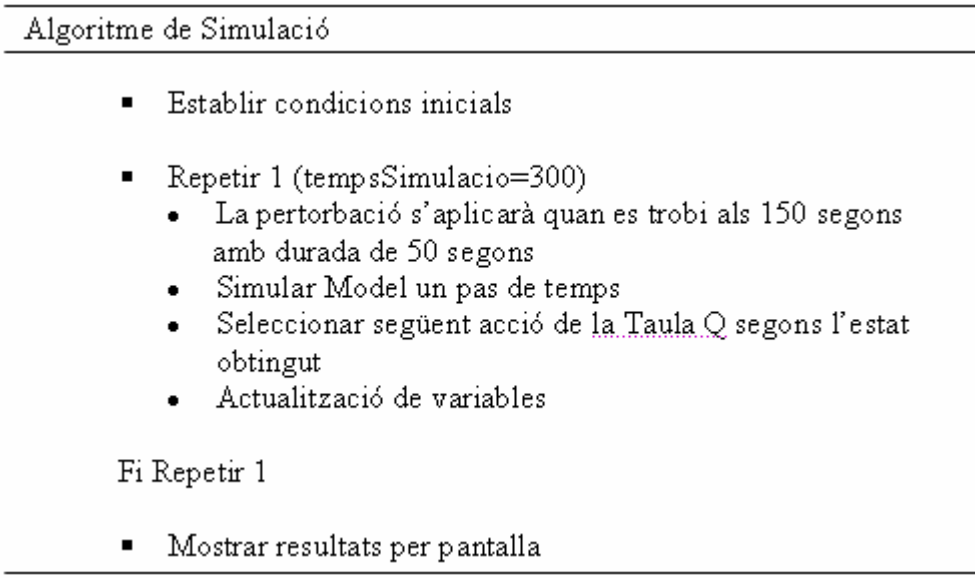


Figura 5.4. *Algoritme de simulació*

Per fer la simulació cal reiniciar les condicions inicials, doncs amb l'algoritme d'aprenentatge queden tots modificats. En aquest cas, en la simulació es parteix d'unes condicions inicials fixes perque en l'aprenentatge es pretenia explorar alternatives mentre que en la simulació es parteix d'unes condicions inicials conegudes.

El temps de simulació és més llarg per poder aplicar una pertorbació, que en aquest cas és l'aplicació d'un graó unitari, a mitjans de la simulació.

Segons els resultats es podrà determinar si l'aprenentatge realitzat permet portar un sistema al nivell desitjat i, a més, veure si és capaç d'absorbir i tractar de forma ràpida la pertorbació.

5.3. Paràmetres d'aprenentatge

Els resultats que s'obtenen en gran part es determinen per la configuració dels paràmetres d'aprenentatge Sarsa, l'alpha (α) i la gamma (γ).

- ***α (learning rate):*** permet establir la importància que es vol donar al que s'està aprenent. És a dir, si α és proper a 0 s'estarà donant prioritat a allò que ja té aprés; si α és proper a 1 s'estarà donant molta importància a allò que s'està aprenent.

- ***γ (discount factor):*** té repercussió en la recompensa que obtindrà la parella estat-acció. Si γ és proper a 0 la recompensa que s'adjudica a la parella només serà "vàlida" per tenir en compte l'instant en que s'ha donat lloc la situació. Si γ és proper a 1 la recompensa que s'adjudica a la parella tindrà repercussió en el futur, és a dir, totes les situacions anteriors que li han donat una puntuació es tenen en compte.

Per calibrar aquests paràmetres per obtenir el millor aprenentatge i conseqüentment un resultat molt bo no hi ha cap norma o estratègia a seguir. Als llibres, que es troben a la bibliografia, no esmenten cap regla o estratègia marcada a seguir que sigui efectiva sinó que es basen en el desig de l'usuari. En base a la descripció anterior l'usuari estipula quins valors són els més adequats.

Després de realitzar proves és molt clar que el millor mètode per calibrar aquests dos paràmetres és l'experiència. Els sistemes són diferents i per tant amb comportaments diferents, i no hi ha una norma a seguir que sigui eficient per qualsevol sistema o problema a tractar. La realització de proves proveeix a l'usuari un coneixement que permet, després de diverses proves, saber quins valors són els que s'ajusten millor pel problema a tractar. Inicialment es parteix més d'una intuïció que no pas una norma.

El coneixement obtingut no és vàlid per tot sinó que serveix per a cada cas concret, és possible que després de tractar diversos problemes una persona sigui capaç de proposar uns valors que s'acostin molt a l'òptim.

Una manera més científica per tractar aquest tema és realitzar un estudi dels paràmetres per cada cas. Es tractaria de realitzar un escombrat dels dos paràmetres estudiant l'error obtingut, a més, els valors a escombrar tenen un abast petit (de 0 a 1) el qual facilita, relativament, la prova. El que pot arribar a ser tediós és la realització de les proves quan el sistema a estudiar és complex i té un aprenentatge molt llarg i costós pel que fa al processador.

Per avaluar una configuració qualsevol dels paràmetres alpha i gamma, com s'ha dit, observar l'error és la millor opció. És difícil saber quan realitzar l'avaluació, doncs per fer-la hi han altres factors que entren en joc com ara el sistema o política de reforç o el nombre d'episodis en l'aprenentatge. Suposant que es té un bon sistema de puntuació i el nombre d'episodis és suficient com per que l'algoritme aprengui, la manera emprada per calcular l'error seria el ECM (Error Quadràtic Mig) (5.1).

$$\mathrm{ECM} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \left(\mathrm{consigna} - \mathrm{sortida}\right)^2 \qquad (5.1)$$

on N=tempsSimulaccio/stepTemps

El valor ECM correspondria a l'error d'un episodi, pel que es tindran tants valors ECM com episodis d'aprenentatge es tingui. Un cop l'aprenentatge s'acaba és fa la mitja dels ECM, i aquest valor servirà per avaluar una configuració dels paràmetres alpha i gamma.

El mètode aplicat al cas 0, més endavant s'explica en què consisteix aquest cas, mostra la forma obtinguda en el gràfic de 3 dimensions en la figura 5.5. D'aquesta no es pot extreure un resultat definitiu perquè la seva finalitat és orientar a l'usuari de quins valors pot utilitzar. A més, s'ha de tenir present que hi ha factors abans esmentats que influeixen en el resultat, per aquesta raó no es pot prendre el resultat com la cerca òptima pels paràmetres d'aprenentatge.

La configuració per realitzar l'estudi és el següent:

Accions = 0:0.01:2

Estats = -1:0.05:2

t = 0.1 s (pas de temps)

temps simulació = 30 s

ε = 0.1 (a cada episodi nou es multiplica aquest valor per 0.5)

episodis = 40

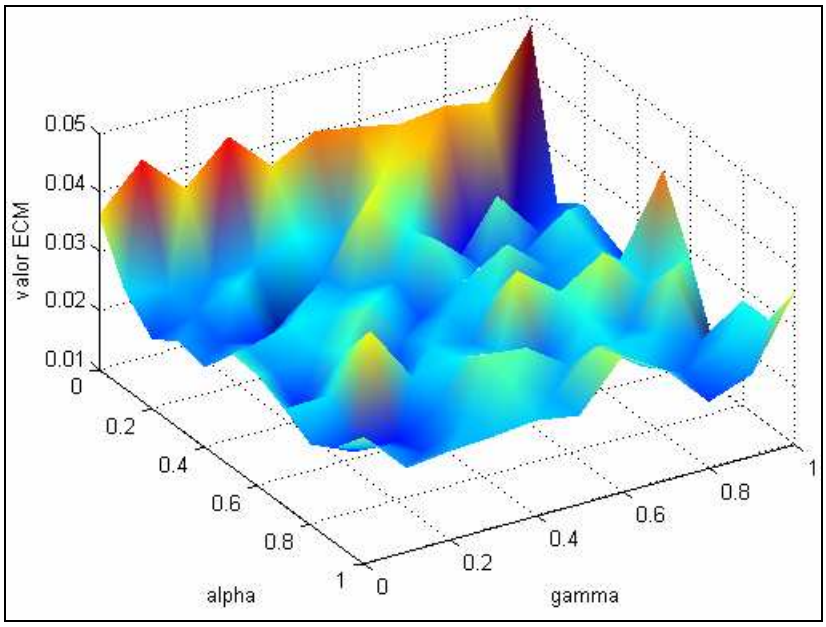


Figura 5.5. *Valor ECM del cas 0 en 3D*

Del resultat s'extreu que l'error en aquest cas és molt petit i per tant es fa difícil poder avaluar-lo. Més endavant es parla del cas 0, en cas de saber més sobre aquest, però cal comentar que els resultats d'aprenentatge obtinguts en aquest són molt bons i conté un error molt petit.

En la següent figura es pot apreciar millor zones que en l'anterior no es poden observar.

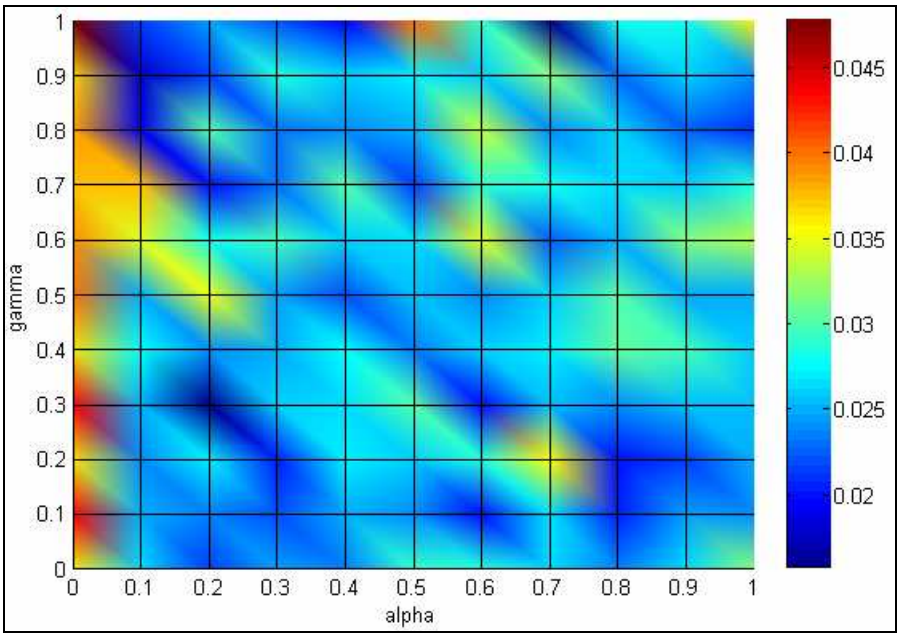


Figura 5.6. *Valor ECM del cas 0 en 2D*

L'última imatge determina que les zones blaves i fosques són les millors configuracions pel cas que es tracta. Tal i com es veu, hi ha diferents configuracions possibles, el que es pot descartar per complet és tenir una alpha amb valor zero, perquè l'aprenentatge tindria molt d'error.

Partint dels resultats anteriors, no es pot concloure quines configuracions obtindran millors resultats degut al color blau que predomina en el gràfic, és per això que per ressaltar més les configuracions possibles dels dos paràmetres es realitza el mateix estudi, però aquest cop disminuint el temps de simulació durant l'aprenentatge. Reduint el temps se centra l'estudi en el primer tram de la simulació, zona on el control tracta d'arribar a la consigna partint des de la condició inicial.

La configuració per realitzar l'estudi és el següent:

Accions = 0:0.01:2

Estats = -1:0.05:2

t = 0.1 s (pas de temps)

temps simulació = 6 s

ε = 0.1 (a cada episodi nou es multiplica aquest valor per 0.5)

episodis = 40

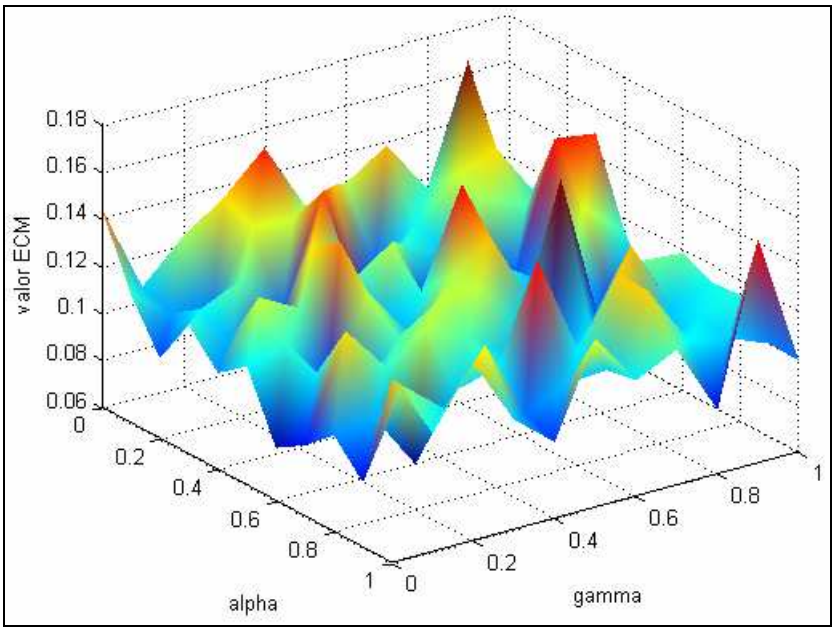


Figura 5.7. *Valor ECM del cas 0 en 3D (amb menor temps de simulació)*

En reduir el temps s'observa en la figura 5.7 com l'error augmenta respecte l'anterior permeten un millor anàlisis sobre tot el conjunt el qual abans era tot d'un mateix color.

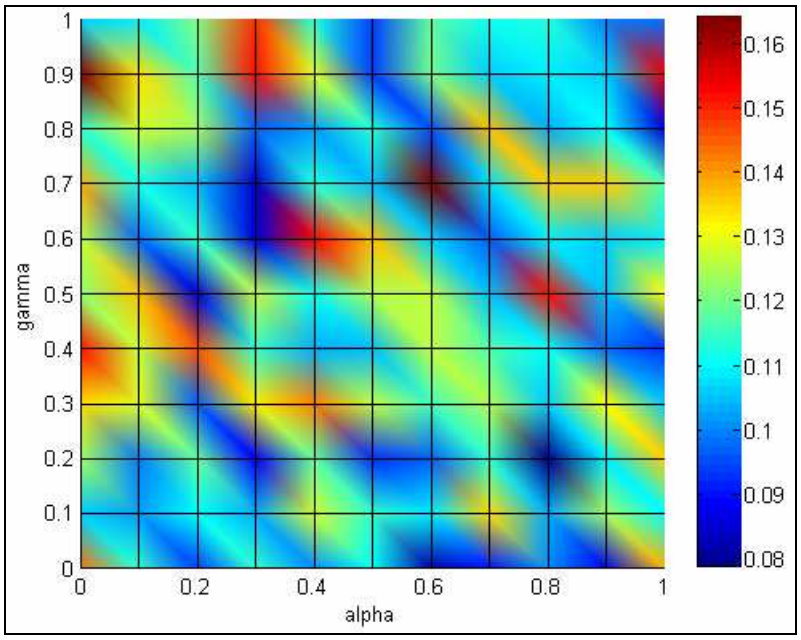


Figura 5.8. *Valor ECM del cas 0 en 2D (amb menor temps de simulació)*

La diferència entre la figura 5.8 i la figura 5.6 és notable, algunes zones blaves fosques que en la figura anterior eren susceptibles de ser una bona configuració ja no ho són tant amb aquesta nova gràfica. Pel contrari, hi ha zones en que es perd l'interès amb aquest segon estudi, on l'error és gran.

La zona interessant, comparant les dues imatges, és la compresa amb un alpha entre 0.5 i 0.9 i amb una gamma entre 0 i 0.3. És la part que més o menys es manté en els dos estudis.

Altre cop, és important recordar que només serveix com a orientació, que el sistema que es tracta, per la seva simplicitat, dóna molt poc error en l'aprenentatge obtenint bons resultats de simulació i per això els valors ECM són molt petits. Sent tot això veritat, les parametritzacions d'alpha i gamma poden augmentar el rang i no limitar-se només a les zones blaves fosques.

Un dels inconvenients d'aquest mètode és la dependència que hi ha d'altres factors que fan impossible establir un estudi d'aquest tipus sinó es realitzen prèviament varies proves per comprovar el bon funcionament de tot el conjunt, el qual implica buscar inicialment de forma intuïtiva uns valors i amb l'evolució de les proves adquirir una experiència que porta a l'usuari a convergir en una configuració amb resultats acceptables.

El temps de processat és un dels inconvenients a tenir en compte, aquest dependrà de la complexitat del model a estudiar doncs pot requerir un pas de simulació molt petit, necessitar un temps de simulació llarg i, a més a més, un nombre d'episodis alt per realitzar l'aprenentatge amb èxits.

En conclusió, la determinació dels dos valors d'aprenentatge és fa molt difícil si es pretén utilitzar alguna eina, mètode o estratègia per convergir en una combinació factible. La solució més immediata, proposada sempre en els llibres i articles trobats, és l'ús de l'experiència.

5.4. Sistema de puntuació

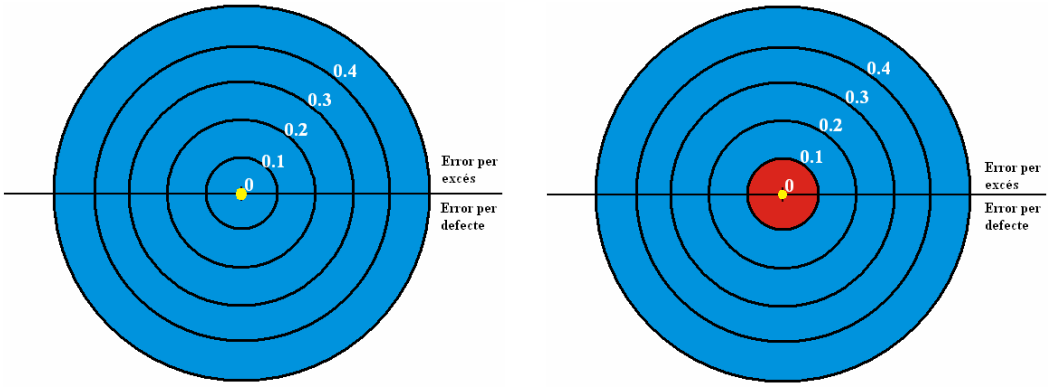
El sistema de puntuació fa referència als reforços, és a dir, quin reforç o puntuació s'ha de donar a una parella estat-acció segons l'error obtingut.

Aquest tema és pot ser el més important de tots perquè determina el bon o mal aprenentatge de l'algoritme. En aquest estudi la manera en que es puntua ha de ser diferent comparant amb els articles, projectes trobats a la xarxa o exercicis proposats en llibres.

La gran majoria, per no dir tots, tracten problemes on un sistema ha d'anar d'un punt A a un punt B, quan arriba al punt B l'aprenentatge s'atura i torna a començar des del punt A reforçant així el camí. El que es diferencia l'estudi de la resta és que no només ha d'aprendre el camí per anar del punt A al punt B, sinó que ha de mantenir-se en el punt B sempre i a ser possible, en cas d'una pertorbació ser capaç de reconduir la situació per acabar en el punt B.

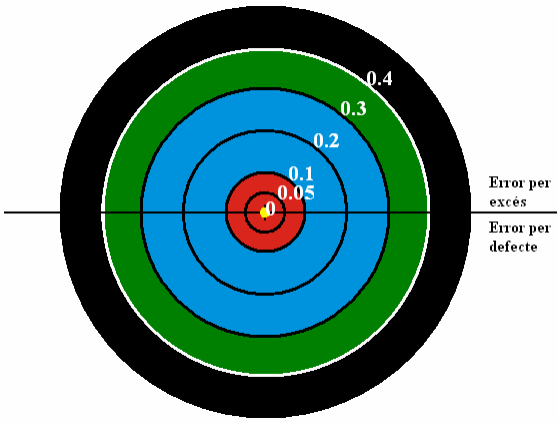
Gran part del temps emprat en aquest estudi s'ha utilitzat per tractar de trobar un sistema de puntuació perquè existeix una relació directa entre la manera de puntuar i el resultat obtingut, no només durant l'aprenentatge sinó també en la simulació del model un cop finalitzat el període d'aprenentatge.

Els reforços que s'aplicaran estan basats en l'estructura 5.1b, per tant es tenen que avaluar parelles d'accions i errors. La millor manera per explicar com puntuar les parelles d'accions-estats és fer un símil amb una diana.



a) Reforçar només l'objectiu

b) Reforçar l'objectiu i un error de 0.1



c) Reforç de l'objectiu i marges d'error

Figura 5.9. *Símil de reforços amb una diana*

La figura 5.9a mostra la política de reforços més utilitzada per aquests tipus d'algoritmes, on només es premiaria que l'error, portat per les accions, fos zero. La idea és de donar una puntuació positiva alta a l'objectiu principal, per exemple 100 punts, i la resta d'estats possibles una puntuació negativa, -1 punt. Aquesta estructura no és gens bona aplicada en aquest estudi, els resultats que s'obtenen són molt dolents. És molt difícil fer que el model aconsegueixi un error zero respecte la consigna, i si ho fa, arriba mitjançant una parella estat-acció en que l'acció que s'aplica porta el sistema a un estat següent molt lluny de l'objectiu.

Per tal de seguir el sistema de puntuació més emprat, en la figura 5.9b es dóna una puntuació alta i positiva per l'objectiu principal, una puntuació mitjana/petita per un error comprés entre el 0 i 0.1, la resta romandria com en l'anterior estructura. Això proporciona una mica més d'estabilitat i es pot veure un millor comportament respecte l'anterior però els resultats no són acceptables.

En la següent estructura, figura 5.9c, es proporciona puntuació positiva entre els rangs d'error fins a 0.5. El rang d'estats (els estats són els errors possibles), serà major que el presentat en la diana, i les puntuacions dins la diana seran positives, de manera esglaonada, i la resta negatives. El que pretén és encarar l'aprenentatge cap a l'objectiu, i aquesta filosofia dóna bons resultats.

Com el sistema anterior només fa que millorar, perquè no utilitzar una funció que adjudiqui una puntuació positiva segons l'error que s'obtingui?. Cal recordar que la puntuació no només es dóna per l'error sinó que també per l'acció que l'ha portat fins aquell estat.

$$r = 300 \cdot e^{\left(-\frac{(error-0)^2}{2 \cdot 0.3^2} \right)} \qquad (5.2)$$

On r és el reforç que es dóna al conjunt estat-acció. La seva representació gràfica amb un rang d'error de -2 fins 2, i de precisió de 0.01 és la següent.

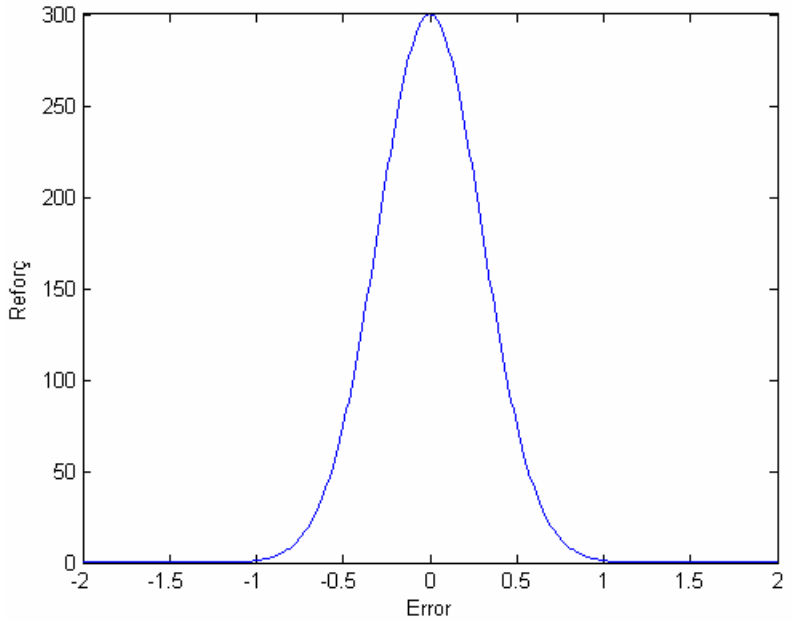


Figura 5.10. *Funció gaussiana com a reforç*

La funció està ajustada de tal manera que amb un error 0 s'obtingui la puntuació màxima de 300 punts mentre que la resta, la seva puntuació decreix.

El problema de la funció és que sempre donarà una puntuació positiva, i en cap moment es penalitzarà que algunes accions portin el sistema, d'un estat molt bo, a un de pitjor. Seria molt bona la funció si el procés de portar el sistema a un error 0 fos el suficientment pautat i un cop arribés a l'objectiu fos capaç de mantenir-se, però no és el cas.

Fent un repàs a totes aquestes opcions es pot dir que en certa manera el comportament que s'obté es correspon, aproximadament, amb un control de tipus proporcional P doncs només es té en compte l'error. Podrien arribar a funcionar, relativament bé, amb sistemes que només necessitessin d'un control P, i tot i així, es tenen dubtes perquè el resultat no només depèn del reforç.

Amb aquesta reflexió es planteja: aprofitant que els estats són l'error, com amb un control PID, es podria utilitzar l'estat per reforçar la

parella estat-acció segons l'error, el qual seria l'acció proporcional P, fer la diferència d'error actual amb l'anterior, fent d'acció derivativa D, i finalment l'error acumulat, com a acció Integral.

La realitat i els resultats diuen que no és tan fàcil com sembla, la idea aproximaria encara més l'algoritme a l'estructura clàssica. La problemàtica es presenta quan amb la diferència d'errors, part D, s'ha d'establir un rang i aquest rang a vegades interessarà que sigui positiu i d'altres negatiu, i per l'altre costat, amb la part integral I és complicat establir un rang i aplicar un reforç. Llavors, assignar les recompenses corresponents es fa més complex del que ja era.

La solució adoptada és la barreja d'un sistema de puntuació com el de la figura 5.9c i la d'un PID. Per una banda es fa una diana com la mostrada en la figura esmentada, sense arribar més lluny que un error de 0.5, i per l'altra banda, el reforç s'assignarà en funció del seu error anterior. Aplicant aquesta combinació s'està utilitzar una estructura PD, perquè es vigila la desviació de l'error respecte de la consigna, que correspondria a l'acció proporcional P, i amb la comparació de l'error anterior s'aconsegueix una acció derivativa D. La filosofia adoptada possiblement no és la millor però aproxima la sortida del model a l'objectiu marcat. El que milloraria la política de reforços seria la introducció de l'acció integral, però no s'ha trobat.

De per sí, és complicat establir una puntuació sigui quin sigui el plantejament de reforços que s'apliqui, però si s'afegeix la complexitat d'escollir els rangs i la dificultat que planteja cada cas converteix l'elecció del sistema de puntuació en ser un tema susceptible d'estudiar a part. Tal és així, que es va buscar articles a la xarxa, concretament a http://ieeexplore.ieee.org, per trobar estudis sobre sistemes de puntuació, fins i tot es va preguntar en el món dels videojocs, món on és habitual utilitzar puntuacions per sistemes de batalles entre altres. La conclusió de la cerca és que no es troben estudis sobre sistemes de puntuació, i en tot cas, s'utilitza el prova i error per balancejar les puntuacions.

5.4.1. Estructura de la taula Q

En la taula Q és on es registra l'aprenentatge i posteriorment descriurà el comportament del model en estudi. Segons com es plantegi afecta al sistema de puntuació, és per això que s'inclou dins l'apartat del sistema de puntuació.

La taula es composa pels estats, per les accions i la puntuació que contingui cada combinació. En apartats anteriors s'ha deixat clar que els estats podrien contenir més informació que l'error, però per qüestions de dificultat en manipulació de dades i l'estructura utilitzada no es creu convenient, en aquest estudi, debatre encara més.

L'estructura utilitzada per l'estudi és similar a l'exemple que s'exposa a continuació.

	Acció1	Acció2	Acció3	Acció4	Acció5	...
Estat1	-5	-6	-4.5	-4	0	
Estat2	-40	-32.3	-51.2	-20.6	-19.8	
Estat3	-12.7	-112.1	-65.3	45.9	-10.9	
Estat4	-24.7	-27.3	-12	-15	-21.1	
...						

Figura 5.11. *Exemple de taula resultant de l'aprenentatge*

S'intueix que la taula pot ser tan gran com es desitgi, en funció de la precisió. Però com més gran sigui més complicat serà ajustar les puntuacions per fer justícia al rang d'estats i accions, i l'aprenentatge serà costós ja que tindrà més opcions i necessitarà o més temps de simulació en el procés d'aprendre o més episodis.

Idealment, suposant que l'aprenentatge fos immediat, sense grans desviacions i el recorregut de la sortida del model seguís en els primers instants de la simulació un camí progressiu cap a l'objectiu, la taula descriuria una puntuació positiva en diagonal cap a l'objectiu zero. Però la realitat és una altra, tal i com es veu en els casos d'estudi.

Recordant les possibles maneres de puntuar emprats als casos, un dels defectes era no tenir una manera d'integrar l'acció I en ella. Manipulant l'estructura de la taula Q es pot aconseguir una estructura que inclou l'acció integral.

En comptes d'establir un rang d'accions, és limita a 3 possibles accions: augmentar, disminuir o mantenir-se.

	Acció1 (0.1)	Acció2 (0)	Acció3 (-0.1)
Estat1	-5	-6	-4.5
Estat2	-40	-32.3	-51.2
Estat3	-12.7	-112.1	-65.3
⋮	⋮	⋮	⋮

Figura 5.12. *Exemple de taula amb 3 accions*

El funcionament consistiria en sumar a l'acció que s'aplica directament al model el valor de l'Acció1, Acció2 o Acció3 (referent a la figura 5.12) augmentant l'acció en cas de no arribar a l'estat d'error 0. En cas d'arribar a l'estat d'error 0 es reforçaria l'Acció2 doncs al no sumar cap valor es mantindria aquell que ha portat el sistema a error zero. Finalment, en cas de passar-se del valor de la consigna, l'Acció3 restaria valor a l'acció aplicada aconseguint una manera d'estabilització. Com s'aprecia, l'acció integral estaria present en aquesta manera d'actuar.

El resultat, altre cop, no és l'esperat i el que s'obté no és millor que l'estructura anterior amb el sistema de puntuació de l'apartat anterior. Augmentar la sensibilitat o afegir un parell d'accions, aconseguint tenir 5 accions possibles, no millora el resultat. Per aquesta raó es retorna a l'estructura anterior vist en la figura 5.11, estructura emprada en els exemples estudiats durant l'estudi.

5.5. Casos de l'estudi

En aquest apartat s'explica el resultat obtingut aplicant les estructures, sistemes, etc, descrits en els apartats anteriors. A més, es realitza una comparativa de l'algoritme Sarsa amb el control PID calculat per cada cas en concret.

En l'estudi s'afegeix un cas el qual no es troba en la Benchmark 2008. La raó per la qual s'afegeix és la d'utilitzar un sistema molt bàsic, basat en el primer cas, per comprovar si la complexitat d'un model és un factor determinant en l'aprenentatge.

5.5.1. Cas 0

La següent funció de transferència té la finalitat de proporcionar un model senzill i bàsic en el qual realitzar proves amb més rapidesa i poder extreure conclusions.

$$G(s) = \frac{1}{s+1} \tag{5.3}$$

Els paràmetres per l'aprenentatge són:

Accions = 0:0.01:2

Estats = -1:0.05:2

t = 0.1 s (pas de temps)

temps simulació = 30 s

α = 0.7

γ = 0.55

ε = 0.1 (a cada episodi nou es multiplica aquest valor per 0.5)

episodis = 40

Dels paràmetres cal destacar l'alpha i la gamma, observant la figura 5.8 on es detalla en colors el grau de desviació que es pot tenir en funció dels dos valors, la configuració utilitzada, segons el gràfic, no és la millor però tampoc és dolenta. Els resultats del conjunt de paràmetres en l'aprenentatge és descriuen en les següents figures.

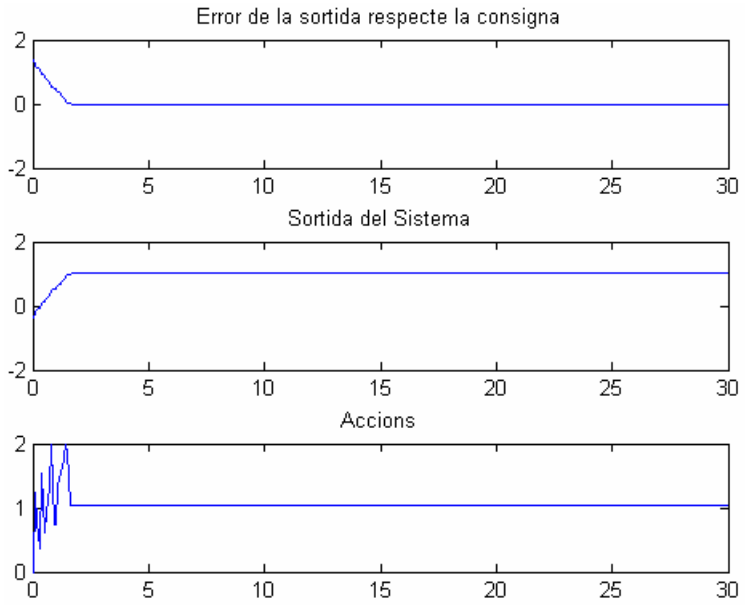


Figura 5.13. *Aprenentatge del cas 0*

La figura 5.13 és l'últim aprenentatge dels 40 que ha realitzat en el procés i en resumeix molt bé com és el seu aprenentatge a excepció, evidentment, dels primers.

L'error obtingut és gairebé zero i l'aproximació cap a l'objectiu el fa de manera progressiva, fent una ullada a les accions preses als 2 segons de simulació es nota que l'algoritme imprimeix unes accions de valors alts per accelerar l'aproximació.

Figura 5.14. *Recompensa total per episodi, cas 0*

La figura 5.14 és representativa de l'aprenentatge realitzat i confirma el que s'ha dit abans, l'aprenentatge és ràpid obtenint puntuacions positives molt bones. És la forma desitjada en tots els casos doncs no només obté bones puntuacions sinó que es manté en una bona puntuació. Això indica que el resultat, com a mínim s'estabilitzarà.

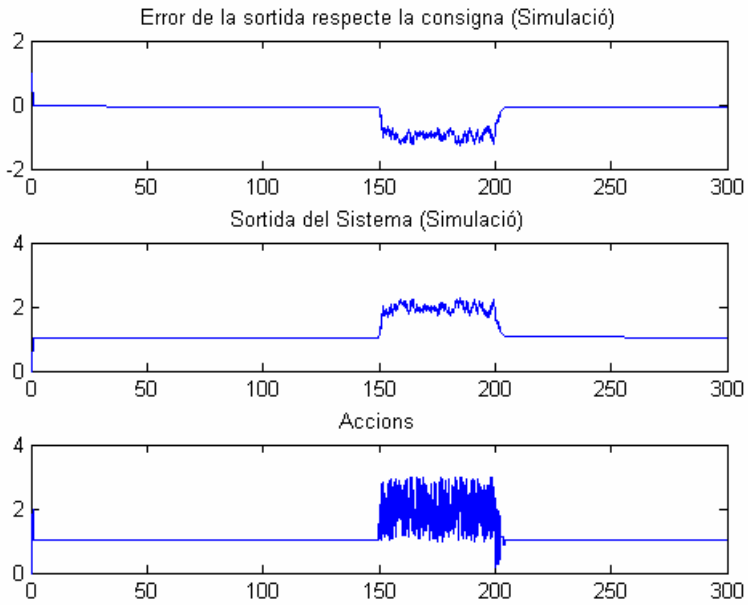


Figura 5.15. *Resultats simulació, cas 0*

El resultat en la simulació, un cop finalitzat el procés d'aprenentatge, concorda amb la figura 5.14 on la puntuació era molt alta i estable. La sortida del model s'estabilitza no en 1 però gairebé i les variacions en la puntuació de les recompenses en gran mesura es deuen a la no adaptació del control davant d'una pertorbació, iniciada als 150 segons de la simulació amb durada de 50 segons.

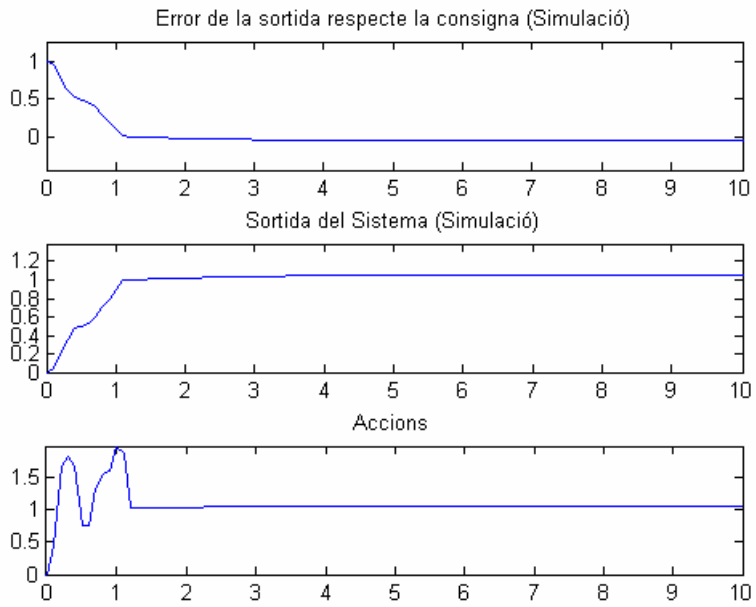


Figura 5.16. *Detall de la simulació, cas 0*

En el detall dels resultats, figura 5.16, durant els 10 primers segons s'aprecia millor l'inici de la simulació i com aquest va disminuint l'error fins gairebé zero.

Els resultats finals són realment bons si s'omet la pertorbació, i el comportament a nivell d'accions preses s'assimila al que podria realitzar un control PID. Cal destacar que és un model molt senzill i que això facilita la feina de l'algoritme per trobar el camí cap a l'objectiu. A més, necessita pocs episodis, menys dels que es parametritzen, per aprendre.

5.5.2. Cas 1

El sistema a simular és el corresponent al (4.1).

Els paràmetres per l'aprenentatge són:

Accions = 0:0.01:2

Estats = -1:0.05:2

t = 0.6 s (pas de temps)

temps simulació = 60 s

α = 0.55

γ = 0.35

ε = 0.1 (a cada episodi nou es multiplica aquest valor per 0.5)

episodis = 500

El valor dels paràmetres es fixen en base al que s'ha observat en les diferents proves realitzades. En general els processos d'aprenentatge realitzat són relativament bons, alguns amb més desviació que d'altres, amb oscil·lació suau al voltant de l'objectiu. La configuració, alpha-gamma, que presenta millors resultats són els que s'han descrit abans vist els resultats de les proves

El procés d'aprenentatge per aquest cas és fa molt llarg doncs necessita, en comparació als altres, de 60 segons de simulació i un número d'episodis alt. La dificultat d'evitar les oscil·lacions que es mostren durant el procés han fet que s'hagin d'augmentar aquests dos valors, a més d'ajustar els paràmetres d'aprenentatge.

Com bé mostra el gràfic de les recompenses total per episodi, figura 5.17, el resultat final no és gaire esperançador doncs no té bones puntuacions.

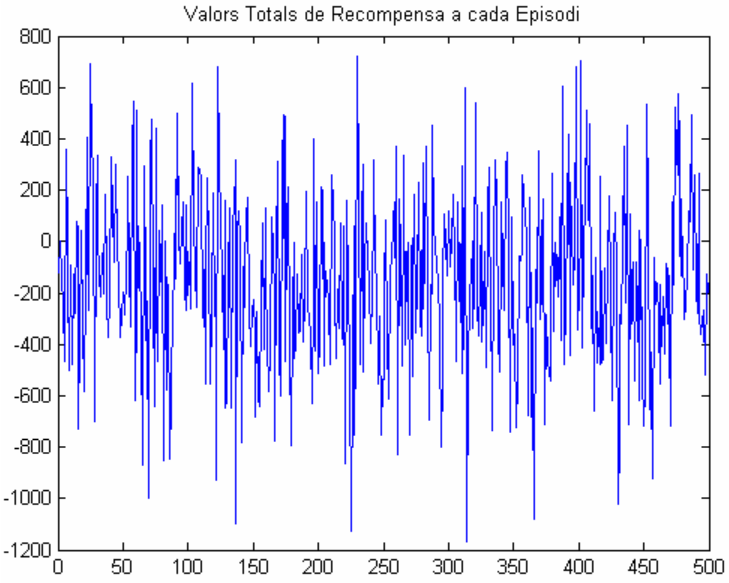


Figura 5.17. *Recompensa total per episodi, cas 1*

La recompensa obtinguda per cada episodi és prou dolenta, en part pels inicis que té perquè normalment acaba superant per excés la consigna i llavors ha de reconduir la situació. Es pot dir que té una puntuació mitja d'uns -100 punts, això no treu que durant l'aprenentatge es pugui observar bones traçades.

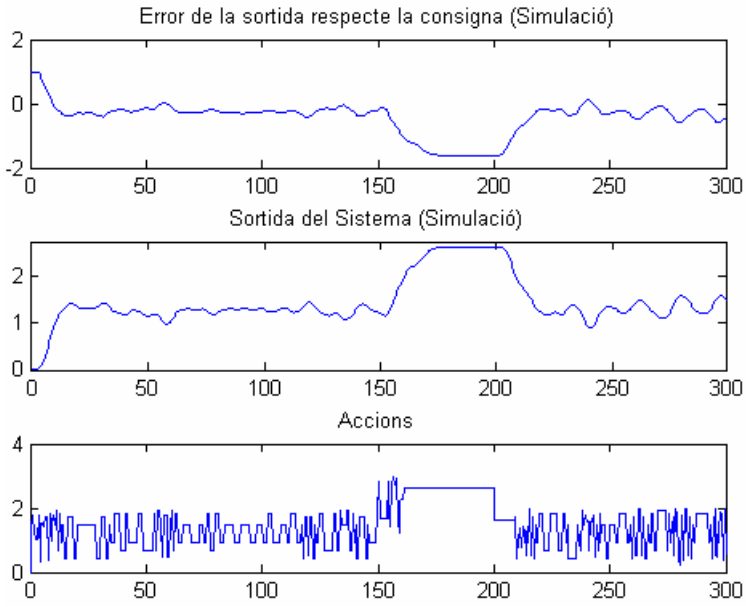


Figura 5.18. *Resultats simulació, cas 1*

En comparació amb el cas 0, l'augment de l'ordre en el cas 1 afecta en el procés d'aprenentatge i conseqüentment en el resultat final com es fa patent en la figura 5.18. Es veu com sembla que les accions actuen però al voltant d'un valor, aproximadament 1.5, observant l'últim gràfic del conjunt.

L'error és considerable i la incapacitat per absorbir la pertorbació és igual que en el cas anterior però en aquest ni tan sols ho intenta, per no oblidar que sembla afectar a la sortida del sistema un cop la pertorbació desapareix.

- COMPARATIVA

La comparació d'ambdós resultats reflexa que els controls de tipus PID són millors. El temps que tarden, en els dos casos, en assolir o sobrepassar la consigna és similar però és molt evident la superioritat del control PID envers l'algoritme aplicat.

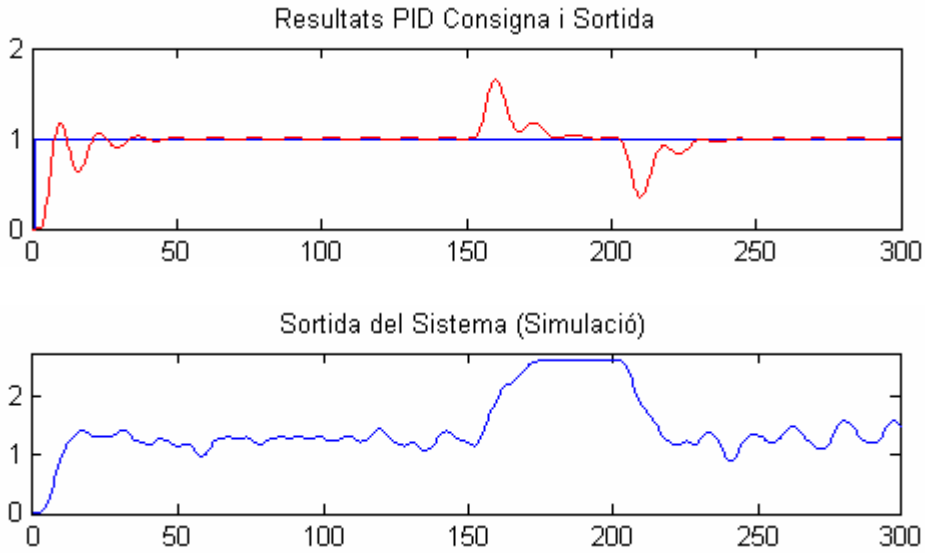


Figura 5.19. *Comparativa de sistemes de control, cas 1*

Mentre un control clàssic elimina les diferències que hi ha entre la consigna i la sortida amb facilitat, l'algoritme Sarsa no pot eliminar l'error i acaba estabilitzant el sistema al voltant de 1.2 amb un error. El control PID, trobat amb la metodologia Ziegler-Nichols, té un sobreimpuls d'aproximadament el 20% del valor de la consigna i

passen 50 segons per estabilitzar el senyal. L'impuls inicial de l'algoritme per portar el senyal al valor de la consigna és sensiblement major, 25% aproximadament, però no pot estabilitzar-ho, deixant un error que va variant.

La capacitat per adaptar-se a una situació imprevista, com és la pertorbació, en el cas del Sarsa, deixa clar que no pot absorbir-la quan amb un PID necessita gairebé 50 segons per adaptar-se. És en aquest punt on tots dos controladors obtenen l'error màxim, per una banda, el control PID té un sobreimpuls generant un error de més 0.7 el qual tarda un temps de 30 segons en estabilitzar el senyal. Per l'altra banda, el Sarsa augmenta l'error que produeix la pertorbació obtenint un error màxim de més 1.2.

Tal i com es fa referència al capítol 5.4, on s'al·ludeix que la política de reforços per la seva estructura s'assimila a un control PD, s'ha calculat un control PD aplicant la metodologia de Cohen-Coon. Els arxius es poden trobar en el document digital de l'estudi.

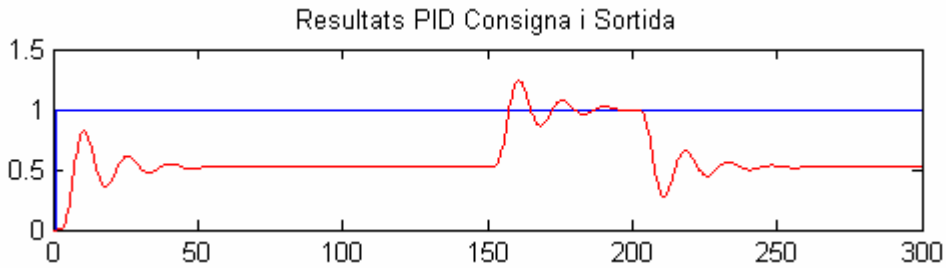


Figura 5.20. Control PD, cas 1

Si es fa la comparativa amb el resultat de l'algoritme, figura 5.19, s'observa que l'algoritme té un millor comportament que no pas amb un control clàssic PD. Un controlador tipus PD no pot controlar el sistema vist el resultat on la sortida no assoleix el valor de la consigna, exceptuant quan s'introdueix la pertorbació i no pot adaptar-se a ella. Per tant, la política de reforços, per la seva configuració, millora el

control PD i segurament part de la millora és degut als factors d'aprenentatge, el sistema de control, etc.

5.5.3. Cas 2

El sistema a simular és el corresponent al (4.3). Per establir el pas de temps de simulació s'aplica el criteri esmentat en l'apartat 5.2.1.

En aquest cas es tarda aproximadament 0.7 segons en assolir el 63% de la resposta, llavors el pas de temps seria de 0.07 segons. El pas de temps calculat per aquest sistema fa que el procés d'aprenentatge i simulació sigui excessivament lent, per aquesta raó s'ha optat per arrodonir el temps a 0.1 segons.

Els paràmetres per l'aprenentatge són:

Accions = 0:0.01:2

Estats = -1:0.05:2

t = 0.1 s (pas de temps)

temps simulació = 30 s

$\alpha = 0.5$

$\gamma = 0.5$

$\varepsilon = 0.1$ (a cada episodi nou es multiplica aquest valor per 0.5)

episodis = 200

El sistema té tres pols, per tant, el seu control hauria de ser relativament fàcil per trobar-se entre els casos zero i u abans tractats. En els anteriors es veu com l'augment de l'ordre afecta en el control.

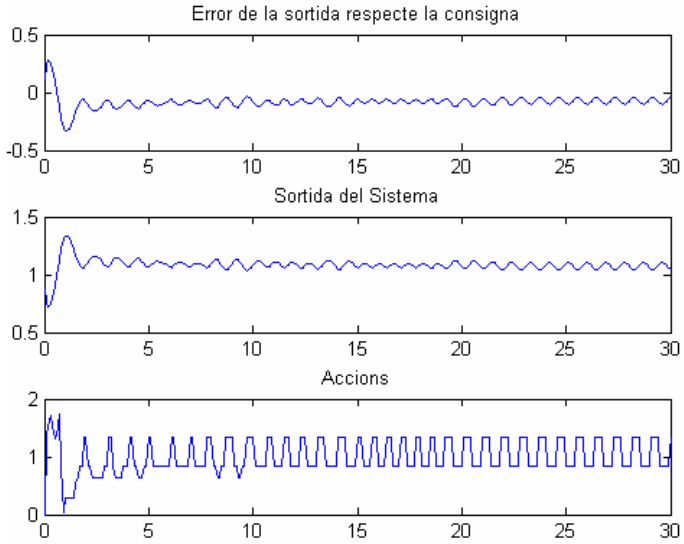


Figura 5.21. *Aprenentatge del cas 2*

Durant el procés d'aprenentatge del model, la seva sortida es va modelant com la mostrada en la figura 5.21. Es pot veure com en les accions preses per l'algoritme hi ha valors que tenen molt bona puntuació, fent que hi hagi una oscil·lació.

Analitzant l'aprenentatge es podria pensar que les recompenses totals per cada episodi pot tenir una forma com en el cas zero.

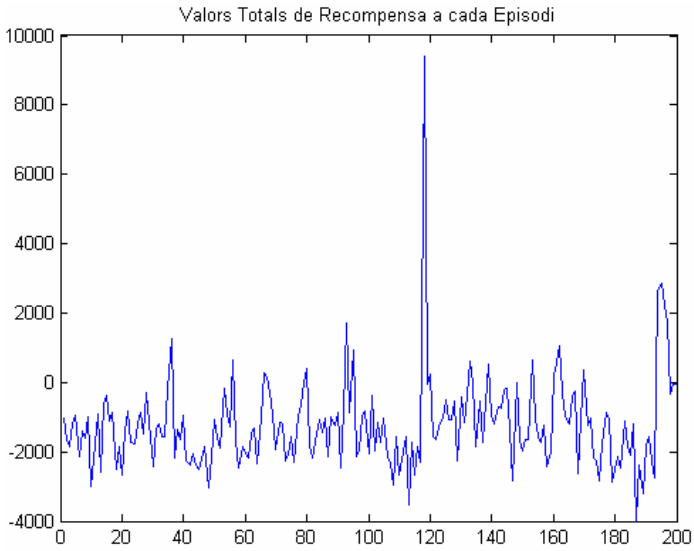


Figura 5.22. *Recompensa total per episodi, cas 2*

Les puntuacions que es guanyen per apropar-se a l'objectiu queden en un segon pla al tenir oscil·lacions a la sortida. És per això que no es pot prendre una decisió clara partint només de la recompensa total de cada episodi, cosa que se sol fer en els exemples trobats durant la documentació per l'estudi.

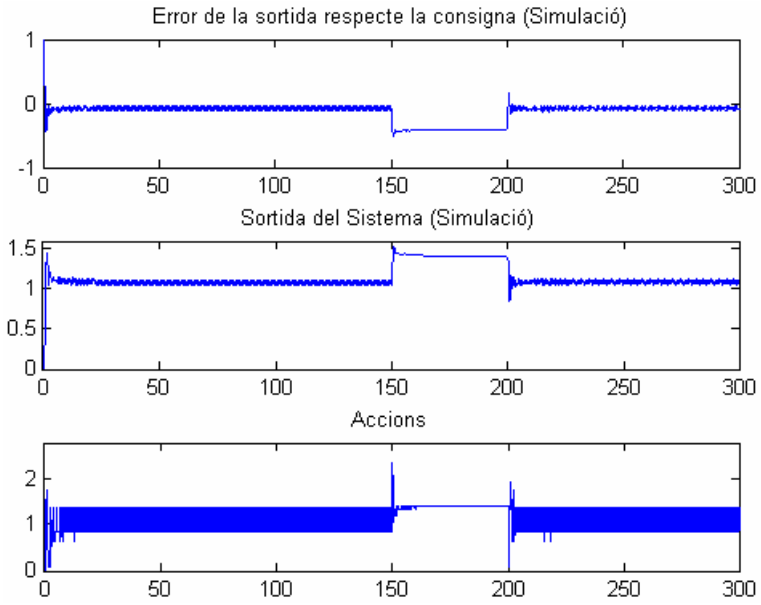


Figura 5.23. *Resultats simulació, cas 2*

L'aprenentatge aplicat al model dóna uns resultats bastant bons, tot i així té la mateixa problemàtica d'adaptació davant una pertorbació que els anteriors casos a més a més de l'error que es pot apreciar. En la següent figura es pot observar amb més detall l'error i les accions preses en la simulació, molt similar al que es veu amb l'aprenentatge.

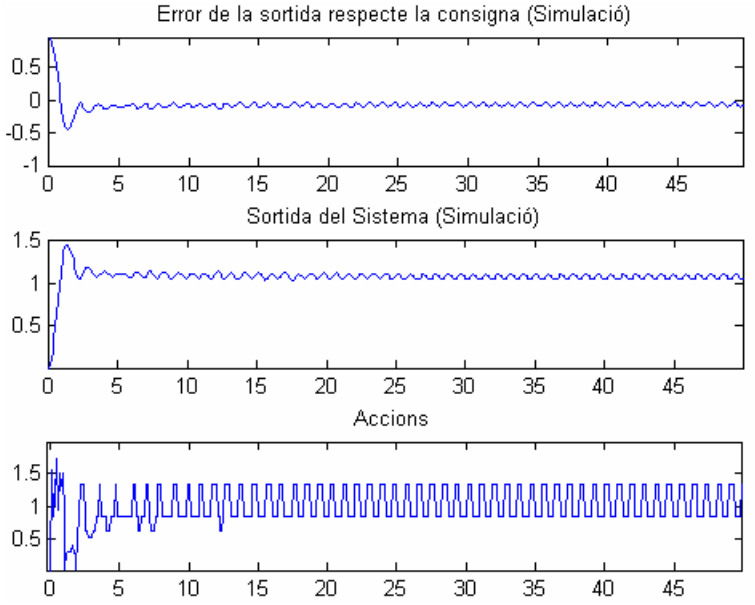


Figura 5.24. *Detall de la simulació, cas 2*

- COMPARATIVA

Valorant els dos controls, l'algoritme té una acció derivativa molt potent perquè porta el sistema amb rapidesa a la consigna, o gairebé, mentre el control PID té una velocitat molt lenta, tan lenta que necessita d'uns 40 segons per estabilitzar la sortida.

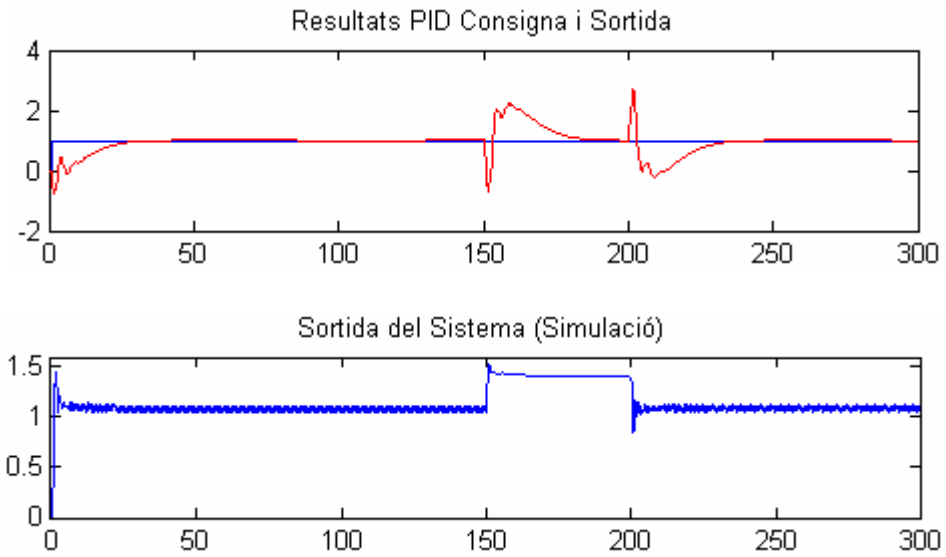


Figura 5.25. *Comparativa de sistemes de control, cas 2*

Per tant, es pot dir que la política de reforços emprada fa un bon ús de la comparativa de l'error actual i l'error anterior. Per altra banda, l'acció proporcional no acaba d'estar ben ajustada en el sistema de puntuació per la desviació d'aproximadament 0.1 en règim permanent. El PID no té cap dificultat en eliminar la desviació i l'error en règim permanent doncs la part PI està millor ajustada que no pas l'algoritme. De fet, es fa patent la falta d'una acció d'integració amb el Sarsa, l'error acumulat es fa difícil de controlar.

Per últim, la pertorbació sembla donar dificultats amb un control clàssic però solventat amb un temps d'estabilització de 40 segons, mentre que l'algoritme només és capaç d'aplicar una acció P sense gaire èxit. El sobreimpuls que provoca l'inici i la finalització de la pertorbació aconsegueix l'error màxim de 1.2 aproximadament, per defecte i per excés respectivament, en el control PID. L'algoritme, mentre apareix la pertorbació es manté constant, capaç d'eliminar part de l'error generat estabilitzant-ho en pocs segons. En aquest model, el que dificulta l'aprenentatge és el zero positiu que conté, si aquest no hi fos els resultats serien bastant similars al cas zero per ser d'un ordre petit.

5.5.4. Cas 3

El sistema a simular és el corresponent al (4.5). Aquest model té un retard pur de 1 segon, Matlab té eines que permeten afegir aquest retard amb molta facilitat el problema és que no funciona per l'estudi. Afegint un retard per software, en l'aprenentatge, no permet que la simulació es realitzi amb normalitat. El pas de temps calculat és de 0.12 segons i utilitzant l'eina de Matlab el temps de pas és massa petit impedint l'algoritme aprendre qualsevol cosa. En el moment en que es configura el pas de temps per sobre de la durada del retard, llavors es pot simular, però no estaria en igualtat de condicions a la resta. Per aquesta raó el retard s'afegirà tal i com amb el càlcul del control PID es fa, mitjançant una simplificació de pade.

Els paràmetres per l'aprenentatge són:

Accions = 0:0.01:2

Estats = -1:0.05:2

t = 0.12 s (pas de temps)

temps simulació = 30 s

$\alpha = 0.6$

$\gamma = 0.65$

$\varepsilon = 0.1$ (a cada episodi nou es multiplica aquest valor per 0.5)

episodis = 100

La dificultat del model ve donada pel retard, si aquest fos tractat amb la seva expressió el control no hauria de tenir grans dificultats però, com s'ha dit abans, el retard es fa mitjançant pade. Aquesta transformació del retard provoca que el model incrementi la dificultat augmentant l'ordre i afegint zeros.

En l'aprenentatge es pot veure que el control tindrà dificultats per manejar el model i portar-lo al punt desitjat. En l'elecció dels paràmetres d'aprenentatge poc es pot fer, només es pot disminuir la visió de futur mitjançant la gamma per que accentuï les recompenses immediates i deixar un aprenentatge, l'alpha, mitjà per tal de reconduir les accions passades i aprendre partint del que ja se sap i del que s'aprèn en el moment. Les accions preses per determinar els paràmetres són fetes a base de conèixer el model i el seu comportament.

Durant el procés d'aprenentatge, observant els gràfics que es mostren després de finalitzar cada episodi, deixa clar que el control no serà gaire bo, i les puntuacions rebudes més aviat seran.

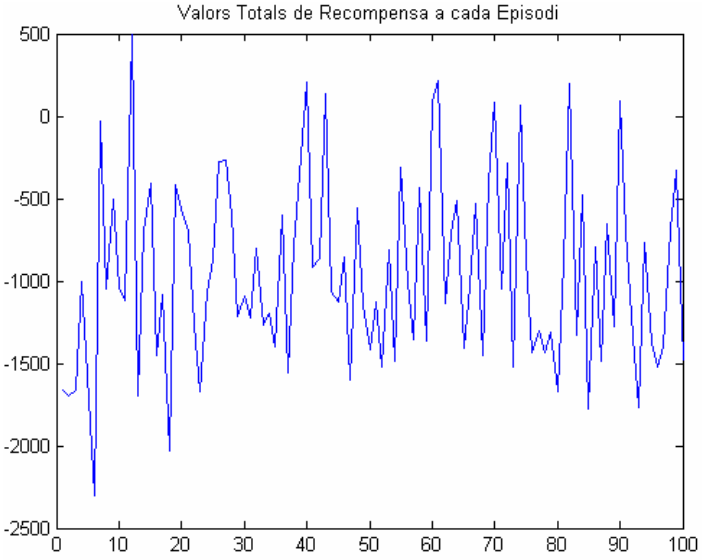


Figura 5.26. *Recompensa total per episodi, cas 2*

Les recompenses rebudes en el procés i valorant-les totes, queden centrades al voltant d'una puntuació de -1000 punts. Com abans s'ha comprovat, no té perquè ser negatiu en el resultat final però aquest fet i afegint que el procés d'aprenentatge no acaba de convergir en l'objectiu de manera satisfactòria, el resultat s'allunyarà del que s'espera.

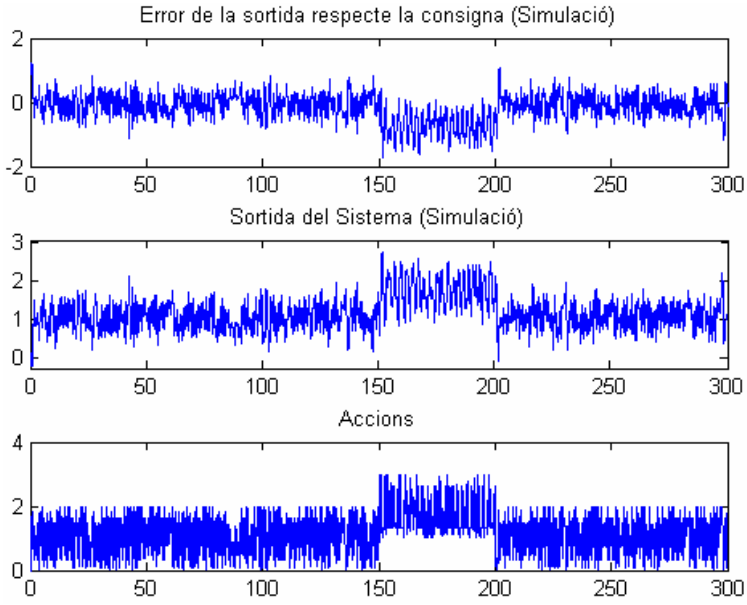


Figura 5.27. *Resultats simulació, cas 3*

La simulació resultant és un conjunt sorollós i del qual difícilment s'obtindran millors resultats.

- COMPARATIVA

La comparativa té per guanyador el control clàssic, capaç d'arribar ràpidament a l'objectiu marcat, eliminant qualsevol mena d'error possible i adaptant-se eficaçment davant una pertorbació. Té un petit sobreimpuls però imperceptible i el temps d'estabilització és aproximadament 10 segons. L'algoritme té una acció derivativa ràpida inicialment però degut a la falta d'una acció que elimini l'error en règim permanent no pot controlar el sistema i obtenint un gran error. És possible que disminuint d'alguna manera l'efecte derivatiu l'error no fos tan gran.

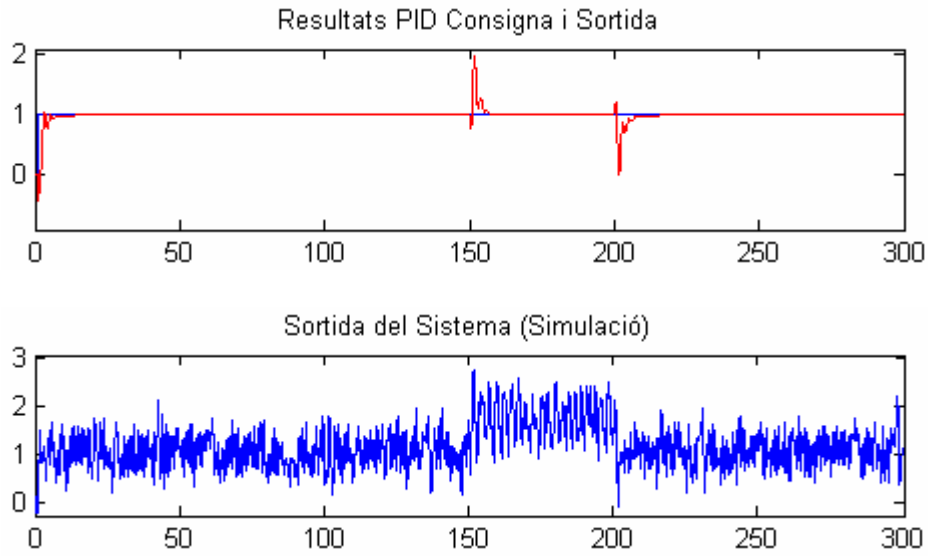


Figura 5.28. *Comparativa de sistemes de control, cas 3*

L'error en règim permanent té visions oposades, el primer té un error zero en règim permanent mentre que l'algoritme conté un error de ±1. En l'únic moment que el control PID conté error és en el sobreimpuls produït a l'inici i final de la pertorbació, el qual l'estabilitza en 10 segons i aconsegueix un error absolut màxim de 1.1 al finalitzar la pertorbació.

El segon controlador es fa difícil distingir l'inici i final ja que el comportament és el mateix que en règim permanent, únicament durant la pertorbació s'aprecia que l'error obté un offset considerable.

Més enllà de l'ús de la transformació de pade pel retard, el qual afecta clarament, un sistema que contingui zeros sempre presentarà grans dificultats per l'algoritme i si aquest és d'un ordre elevat, les previsions d'un bon control disminuiran.

5.5.5. Cas 4

El sistema a simular és el corresponent al (4.9). El temps que tarda en arribar al 63% és de 0.775 segons, pel que el pas de temps hauria de ser de 0.0775 segons però s'ha optat per arrodonir a 0.08 segons.

Els paràmetres per l'aprenentatge són:

Accions = 0:0.01:2

Estats = -1:0.05:2

t = 0.08 s (pas de temps)

temps simulació = 30 s

$\alpha = 0.5$

$\gamma = 0.45$

$\varepsilon = 0.1$ (a cada episodi nou es multiplica aquest valor per 0.5)

episodis = 80

El model a estudiar té component oscil·latòria, per això els paràmetres d'aprenentatge són més propers a 0, per focalitzar el procés en allò més immediat i poder reaccionar a temps. Durant l'aprenentatge queda patent que no és gens fàcil controlar un model com el cas 4. L'algoritme d'aprenentatge centra el senyal de sortida en la consigna marcada però no pot dominar la tendència oscil·latòria, això provocarà que la taula Q resultant necessària per la posterior aplicació en la simulació contingui error.

Com és d'esperar, les puntuacions rebudes són molt dolentes però no difereixen de la resta de casos tractats a excepció del cas 0. Tan aprenentatge com puntuació semblen anar en consonància, igual que en el cas anterior.

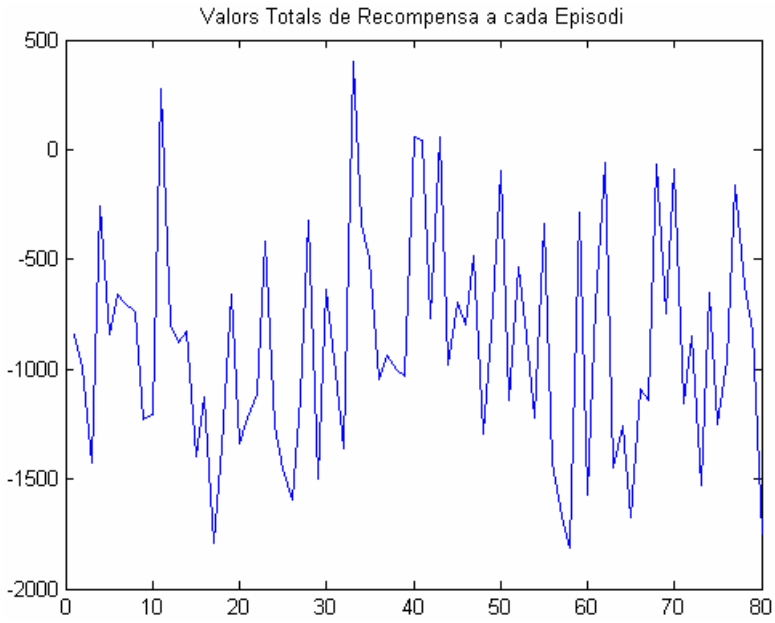


Figura 5.29. *Recompensa total per episodi, cas 4*

El resultat final no sorprèn havent vist el procés d'aprenentatge es podia intuir un resultat com el mostrat, figura 5.30. Intentar ajustar els paràmetres per eliminar l'error és complicat, un augment de la gamma, que premia la visió de futur o lo immediat, només garanteix un augment de l'error perquè aquest s'acumula i té més presència en avaluacions futures. El factor d'aprenentatge va a remolc de la gamma, no es pot aprendre tot, ni tampoc res per ser dolent el resultat.

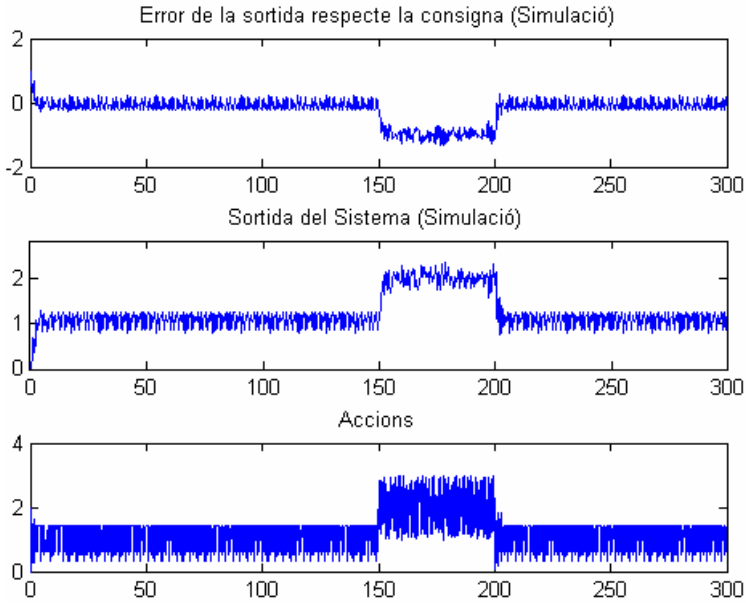


Figura 5.30. *Resultats simulació, cas 4*

- COMPARATIVA

La situació és molt similar al cas anterior, l'error proporcional a la consigna s'elimina, però no l'error acumulat en règim permanent.

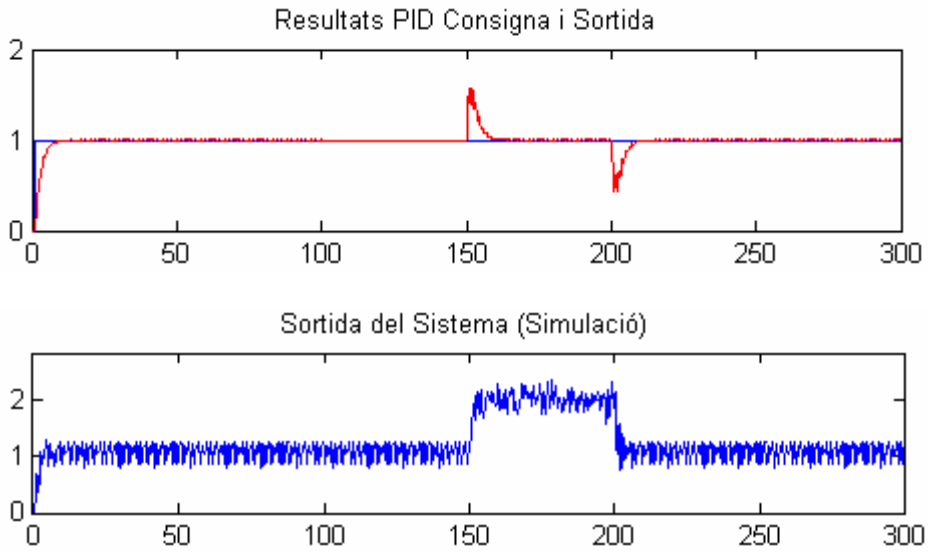


Figura 5.31. *Comparativa de sistemes de control, cas 4*

La comparació visual dels dos tipus de control deixa clara l'opció que s'hauria de triar, els dos no necessiten molt de temps per igualar-se a la consigna, un temps d'estabilització de 10 segons. L'error en règim permanent és de zero en el primer cas mentre en el segon cas s'observa un error de ±0.2. A més, l'adaptació davant senyals imprevistes amb un PID, l'acció és molt ràpida amb un sobreimpuls, tan al principi com al final de la pertorbació, del 60% respecte la consigna esmorteït en 10 segons, aproximadament. El segon control no pot estabilitzar la sortida davant aquest senyal cosa que dóna lloc a la zona de major error.

Com en el cas 1, per aquest cas també s'ha calculat un controlador tipus PD pel sistema amb la metodologia Cohen-Coon que es troba en la documentació digital de l'estudi.

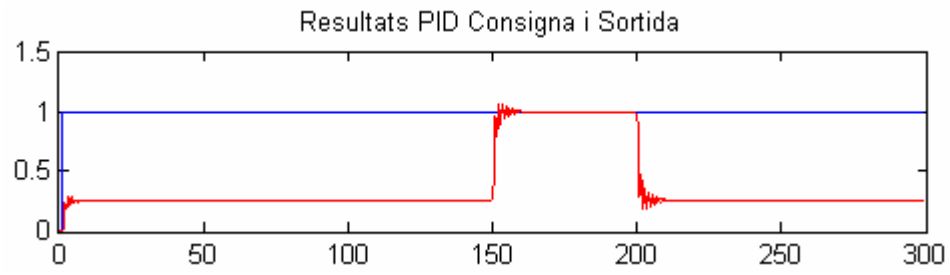


Figura 5.32. *Control PD, cas 4*

L'algoritme torna a demostrar un millor comportament davant un PD ja que porta el model a la consigna, amb un error, mentre el control PD, figura 5.32, té les mateixes dificultats que el control PD del cas 1, figura 5.20.

El sistema no pot funcionar amb un control PD, com reflexa l'últim gràfic, però el Sarsa sí que és capaç de fer-ho, només és necessari ajustar encara més els paràmetres i estructures estudiades abans per tenir resultats amb menor error.

CAPÍTOL 6: CONCLUSIONS

La gran majoria de maquinària o processos que necessiten un sistema de control solen tenir un de tipus PID, o derivat. Evidentment hi han que només amb la supervisió visual o amb un limitador, de tipus mecànic o software, el procés o màquina queda controlat. L'objectiu d'aquest estudi consistia en substituir aquest control tipus PID, o derivat, per un algoritme de caire intel·ligent el qual és capaç d'aprendre per si sol.

En l'estudi queda reflectit en tot moment, ja sigui en els casos o en les parts que component l'algoritme, que un algoritme d'aprenentatge per reforç, el Sarsa en aquest cas, no és tan senzill d'utilitzar. Mentre un control clàssic com és sintonitzar el PID està molt estudiat i tabulat per obtenir un control de manera immediata, un algoritme d'aprenentatge per reforç necessita calibrar molts paràmetres i, tot i així, no aconsegueix els mateixos resultats.

Aquests algoritmes requereixen de més informació per realitzar l'ajust del controlador, i sobretot l'experiència d'aquell qui tracta el propi algoritme. La necessitat de més informació, respecte la sintonització

d'un PID, concorda amb la dependència que existeix amb que aquesta informació estigui ben definida.

La part més important és la definició del sistema de puntuació de la qual no es coneix l'existència de cap metodologia per puntuar, per tant es depèn de l'opinió de l'usuari i de la seva experiència obtinguda del treball amb aquests tipus d'algoritmes. De les qualitats esmentades es determina l'aprenentatge i el posterior resultat aplicant el que s'ha aprés.

L'estructura de control, l'estructura de la taula d'aprenentatge (Taula Q) i la complexitat del model són el següent esglaó en l'impediment d'un bon aprenentatge. Els models no es poden escollir en la vida real sinó que venen donats o s'han de trobar, si es pot, per tant, el bon funcionament queda definit pel propi algoritme.

La configuració dels paràmetres del propi algoritme, alpha i gamma, és costós doncs altre cop depèn de l'experiència tot i tractar d'establir una relació entre l'error obtingut en el procés d'aprenentatge i els dos paràmetres. A més a més, cal afegir que amb una mateixa configuració s'obtenen diferents aprenentatges i, per tant, els resultats seran diferents, no molt distants els uns dels altres però amb una diferència que deixa el dubte sobre la configuració establerta.

Els resultats obtinguts al llarg de l'estudi demostren que, actualment, l'aplicació d'un algoritme d'aprenentatge per reforç no és viable pel temps emprat en la seva configuració i ajust, i tampoc pels propis resultats que no s'acosten al que pot fer un control clàssic.

6.1. Accions futures

Per tal de poder obtenir millors resultats caldria estudiar la utilització de les funcions d'aproximació [2], les quals permetrien donar més informació a l'algoritme i així poder actuar amb més coneixement de causa.

Seria bo trobar alguna relació que permetés l'usuari determinar la configuració òptima d'aprenentatge a fi i efecte que aquest veiés reduït el temps dedicat en ell, així com també la relació del nombre d'episodis necessaris per aprendre.

Per altra banda, un dels grans punts d'estudiats és la política o sistema de puntuació, seria convenient realitzar un estudi sobre com puntuar un resultat, o conjunt en aquest cas, doncs trobar una puntuació que reflecteixi el bon o mal comportament és complicat. És fonamental saber-ho fer perquè el resultat final va directament lligat a aquesta part de l'algoritme.

Finalment, com s'explica al principi de l'estudi hi ha altres algoritmes que podrien ser utilitzats per la finalitat aquí estudiada amb el Sarsa. L'estudi d'altres algoritmes, similars al Sarsa pot aportar millores a aquest.

CAPÍTOL 7: BIBLIOGRAFIA

7.1. Llibres

[1] Sutton, Richard S. i Andrew G. Barto. 1998. Reinforcement Learning: An Introduction. MIT:The Mit Press.

[2] Lucian Busoniu, Robert Babuska, Bart De Schutter i Damien Ernst. 2010. Reinforcement Learning and Dynamic Programming Using Function Approximators. CRC Press.

[3] Passino, Kevin M. 2005. Biomimicry for Optimization, Control and automation. London: Springer-Verlag.

[4] Ogata, Katsushiko. 1999. Ingeniería de Control Moderna, 3a. Ed. Prentice Hall.

7.2. Articles

[5] Joseba Quevedo, Teresa Escobet, Ramon Comasolivas i Albert Masip. 2008. "Benchmark de sistemas para controladores PID". Terrassa: Universitat Politècnica de Catalunya.

[6] F. Morilla, J. Jiménez i F. Vàzquez. 2008. "Propuesta del grupo de la Universidad de Córdoba y la UNED al Benchmark de Controladores PID". Universidad de Córdoba.

[7] V. M. Alfaro, O. Arrieta i R. Vilanova. 2008. "Concurso Benchmark de sistemas para controladores PID". Universidad de Costa Rica i Universitat Autònoma de Barcelona.

7.3. Webs

[8] Cerca d'articles científics. Accés Novembre 2011. http://ieeexplore.ieee.org.

[9] Asociación Española de Inteligencia Artificial (AEPIA). http://www.aepia.org/.

[10] Jose Antonio Martin H. "Personal Page" Accés Agost 2011. http://www.dacya.ucm.es/jam/.

[11] Tim Eden, Anthony Knittel i Raphael van Uffelen. "Reinforcement Learning". Accés 19 Desembre 2011. http://www.cse.unsw.edu.au/~cs9417ml/RL1/index.html.

[12] Luís Rincón. "Propiedad de Markov". Accés Agost 2011. http://es.scribd.com/doc/54960400/7/Propiedad-de-Markov.

[13] "Exemple Codi Q-learning". Accés Agost 2011. http://pastebin.com/15m4qTqk.