ScienceDirect
Disponible en www.sciencedirect.com
Revista Iberoamericana de Automática e Informática industrial 11 (2014) 179–190
www.elsevier.es/RIAI

Controladores multivariables para un vehículo autónomo terrestre: Comparación basada en la fiabilidad del software

Norberto Cañas[a,*], Wilmar Hernández[b], Gabriel González[c], Oleg Sergiyenko[d]

[a] *Universidad Politécnica de Madrid, EUI-Departamento de Informática Aplicada, Ctra. Valencia Km. 7, 28031 Madrid, España*
[b] *Universidad Politécnica de Madrid, EUITT-Departamento de Ingeniería de Circuitos y Sistemas, Ctra. Valencia Km. 7, 28031 Madrid, España*
[c] *Universidad Carlos III de Madrid, Departamento de Informática, 28911 Leganés, Madrid, España*
[d] *Universidad Autónoma de Baja California, Instituto de Ingeniería, BC 21280 Mexicali, México*

Resumen

Se presenta en este artículo la comparación de tres controladores de velocidad (regulador cuadrático lineal -LQR-, proporcional integral derivativo -PID- y borroso) con la intención de determinar cuál de ellos ofrece mejor fiabilidad desde una perspectiva software. Para realizar las pruebas necesarias se utilizaron versiones mutantes de controladores bien ajustados, en los que se inyectaron defectos que simulaban errores de programación. Los controladores fueron diseñados para operar un vehículo autónomo terrestre y fueron ajustados por medio de un algoritmo genético.

Dado el elevado número de pruebas a efectuar se decidió construir un simulador multicomputador con el que se realizaron más de 90000 ensayos. En cada uno de los ensayos se sometió a cada controlador mutante a la realización de un recorrido, de unos 20 minutos de duración máxima, sobre un suelo ligeramente ondulado. Con los datos obtenidos se generaron las curvas de fiabilidad por el procedimiento de Kaplan-Meier, lo cual permitió la comparación de controladores objetivo del estudio.

De las curvas de fiabilidad del software obtenidas se deduce que, en las condiciones experimentales planteadas, el controlador LQR ofrece el mejor comportamiento, el segundo lugar le corresponde al controlador PID y el tercero al controlador borroso.

Palabras Clave:
Fiabilidad del software, robots móviles autónomos, simuladores, método de control LQR, controlador PID, control borroso.

1. Introducción

Los defectos en el software pueden deteriorar el correcto comportamiento de un robot. Por desgracia, no es siempre posible detectar todas las deficiencias existentes en el software antes de ponerlo en explotación (Smidts et al., 2002; Cheng y Merkel, 2008) y lo mismo ocurre en el ámbito de los vehículos autónomos terrestres (Carlson y Murphy, 2005). Una medida de calidad, relacionada con la existencia de defectos en los programas, es la fiabilidad del software, concepto que se define como "la probabilidad de que el software no cause el fallo de un sistema por un periodo de tiempo especificado y bajo condiciones también especificadas" (IEEE-Reliability-Society, 2008).

En relación con lo anterior, cuando existen varias opciones de diseño, es una buena práctica de ingeniería evaluar el impacto de las mismas en la integridad y prestaciones del sistema (Short et al., 2008). En este contexto, la finalidad del presente trabajo es contrastar tres tipos de controladores multivariables, evaluando el impacto de los defectos de programación en su fiabilidad, permitiendo con ello utilizar este parámetro a la hora de seleccionar alternativas constructivas.

Dado el elevado número de ensayos a realizar para obtener un aceptable nivel de confianza, ha sido preciso construir un simulador multicomputador del vehículo autónomo terrestre (VAT). Este procedimiento, que permite reducir el tiempo de la fase experimental de años a semanas, está totalmente respaldado por el relevante papel que la simulación tiene hoy en día dentro del ciclo de vida de los sistemas (Sánchez-Peña y Sznaier, 1998; Henry et al., 2003; Arena et al., 2005; Isermann, 2008; Raffo et al., 2009).

El artículo está organizado de la siguiente manera: En la sección 2 se presenta de forma resumida el modelo del VAT utilizado. En la sección 3 se presenta la formulación de los controladores multivariables comparados. En la sección 4 se plantea el plan de trabajo experimental. En la sección 5 se explica la arquitectura del simulador multicomputador construido. En

*Autor en correspondencia. Tel.: +34 91 3367859; fax: +34 91 3367527
Correos electrónicos: norberto@eui.upm.es (Norberto Cañas), whernan@ics.upm.es (Wilmar Hernández), gabriel@arcos.inf.uc3m.es (Gabriel González), srgnk@iing.mxl.uabc.mx (Oleg Sergiyenko)

http://dx.doi.org/10.1016/j.riai.2014.02.002

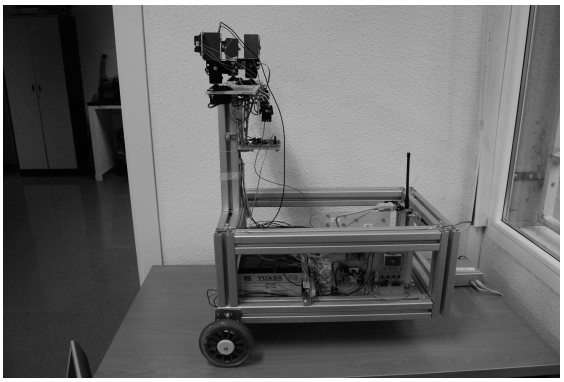

Figura 1: VAT desarrollado para realizar actividades docentes y de investigación

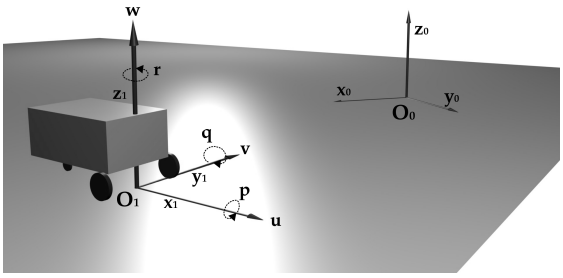

Figura 2: Representación de los marcos de referencia local y global.

la sección 6 se ofrecen los resultados experimentales y por último, en la sección 7 se exponen las conclusiones.

2. Modelo de VAT

En este apartado se presenta el modelo de robot utilizado, el cual se corresponde con un prototipo desarrollado con fines didácticos y de investigación (Figura 1).

Una explicación moderadamente detallada de cómo se han obtenido las ecuaciones que describen el comportamiento del robot pueden encontrarse en Hernandez and Canas (2011). De forma complementaria, puede utilizarse la referencia (Fossen, 1999) que, aunque especializada en vehículos marinos, ofrece en sus capítulos introductorios información de mucho interés acerca del modelado matemático de dispositivos móviles.

Con la intención de gestionar correctamente la posición y orientación de un dispositivo móvil, es frecuente en robótica utilizar dos marcos de referencia definidos en un espacio afín euclídeo (Figura 2). El primero, $M_0(O_0, x_0, y_0, z_0)$, es un marco de referencia global el cual se ubica en una posición fija. El segundo, $M_1(O_1, x_1, y_1, z_1)$, es un marco de referencia ubicado en el robot y por tanto es móvil. De forma añadida, posición, orientación, velocidades, fuerzas y momentos se identifican según (1), en donde, basándose fundamentalmente en la convención de nomenclatura propuesta en Hydromechanics-Subcommittee (1950), los vectores indicados tienen el siguiente significado:

- $\boldsymbol{\eta_1}$ es el vector que indica en M_0 las coordenadas del origen O_1 de M_1.
- $\boldsymbol{\eta_2}$ es el vector de ángulos que indica la orientación del VAT en M_0.
- $\boldsymbol{\nu_1}$ es el vector de velocidades longitudinales del VAT en M_1.
- $\boldsymbol{\nu_2}$ es el vector de velocidades angulares del VAT en M_1.
- $\boldsymbol{\tau_1}$ es el vector de fuerzas que actúan en el VAT en M_1.
- $\boldsymbol{\tau_2}$ es el vector de momentos angulares que actúan en el VAT en M_1.

$$\begin{array}{lll} \boldsymbol{\eta_1} = [x, y, z]^T & \boldsymbol{\eta_2} = [\phi, \theta, \psi]^T & \boldsymbol{\nu_1} = [u, v, w]^T \\ \boldsymbol{\nu_2} = [p, q, r]^T & \boldsymbol{\tau_1} = [X, Y, Z]^T & \boldsymbol{\tau_2} = [K, M, N]^T \\ \boldsymbol{\eta} = [\boldsymbol{\eta_1}^T, \boldsymbol{\eta_2}^T]^T & \boldsymbol{\nu} = [\boldsymbol{\nu_1}^T, \boldsymbol{\nu_2}^T]^T & \boldsymbol{\tau} = [\boldsymbol{\tau_1}^T, \boldsymbol{\tau_2}^T]^T \end{array} \quad (1)$$

2.1. Ecuaciones dinámicas

Considerando las restricciones no holonómicas del VAT utilizado $\left(\frac{dy}{dt} = \boldsymbol{\nu}(t) = 0, \frac{dz}{dt} = \boldsymbol{w}(t) = 0\right)$, la posición neutral del componente y_G del centro de gravedad (x_G, y_G, z_G), la simetría del cuerpo del VAT, que anulan algunos elementos del tensor de inercia $I\left(I_{xy} = 0, I_{yz} = 0\right)$ y considerando que las velocidades angulares p y q son fundamentalmente impuestas por los cambios de inclinación del suelo, es posible alcanzar las ecuaciones (2) en las que la masa del VAT se denomina m, $\boldsymbol{\tau_M}$ es el vector fuerza generado por los motores de corriente continua, $\boldsymbol{\tau_G}$ es el vector fuerza generado por la gravedad (que puede generar momentos cuando el robot está inclinado), $\boldsymbol{\tau_{fric}}$ es el vector fuerza generado por fricciones y el vector $\boldsymbol{\nu}$ declarado en (1) pasa a ser un vector de solo dos componentes $[u, r]^T$. Una descripción más detallada de $\boldsymbol{\tau_M}$, $\boldsymbol{\tau_G}$ y $\boldsymbol{\tau_{fric}}$ puede encontrarse en Hernandez and Canas (2011).

$$\begin{aligned} \dot{\boldsymbol{\nu}} = & -\boldsymbol{H}^{-1}\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{H}^{-1}\boldsymbol{\Omega}\dot{\boldsymbol{\rho}} - \boldsymbol{H}^{-1}\boldsymbol{\Xi}\boldsymbol{\rho} + \boldsymbol{H}^{-1}\boldsymbol{\tau_M} \\ & + \boldsymbol{H}^{-1}\boldsymbol{\tau_G} - \boldsymbol{H}^{-1}\boldsymbol{\tau_{fric}} \end{aligned}$$

donde

$$\begin{aligned} \boldsymbol{H} &= \begin{bmatrix} m & 0 \\ 0 & I_z \end{bmatrix} \boldsymbol{C} = \begin{bmatrix} 0 & -mx_G r \\ mx_G r & 0 \end{bmatrix} \boldsymbol{\nu} = \begin{bmatrix} u \\ r \end{bmatrix} \\ \boldsymbol{\Omega} &= \begin{bmatrix} 0 & mz_G \\ I_{zx} & 0 \end{bmatrix} \boldsymbol{\Xi} = \begin{bmatrix} mz_G r & -mx_G q \\ I_y q & -I_{xz} r - I_x p \end{bmatrix} \boldsymbol{\rho} = \begin{bmatrix} p \\ q \end{bmatrix} \end{aligned} \quad (2)$$

2.2. Modelo en tiempo discreto

Como se indicó en la introducción, se ha construido un simulador para poder realizar un elevado número de experimentos. Por ello ha sido necesario obtener el modelo en tiempo discreto del VAT, el cual se recoge en las ecuaciones (3 y 4), las cuales se han obtenido a partir de (2).

$$\begin{aligned} \boldsymbol{\nu}(t_{k+1}) = & \boldsymbol{\Phi}(t_k)\boldsymbol{\nu}(t_k) + \boldsymbol{\Gamma}_{\dot{\rho}}(t_k)\dot{\boldsymbol{\rho}}(t_k) + \\ & \boldsymbol{\Gamma}_{\boldsymbol{\rho}}(t_k)\boldsymbol{\rho}(t_k) + \boldsymbol{\Gamma}_{\boldsymbol{M}}(t_k)\boldsymbol{\tau_M}(t_k) + \\ & \boldsymbol{\Gamma}_{\boldsymbol{G}}(t_k)\boldsymbol{\tau_G}(t_k) + \boldsymbol{\Gamma}_{\boldsymbol{fric}}(t_k)\left(-\boldsymbol{\tau_{fric}}(t_k)\right) \end{aligned} \quad (3)$$

$$
\begin{aligned}
\boldsymbol{\Phi} &= e^{-\boldsymbol{H}^{-1}\boldsymbol{C}(\boldsymbol{\nu}(t_k))T} \\
\boldsymbol{\Gamma}_{\dot{\boldsymbol{\rho}}} &= \left[\int_{t_k}^{t_{k+1}} e^{-\boldsymbol{H}^{-1}\boldsymbol{C}(\boldsymbol{\nu}(t_k))\left(t_{t_k+1}\right)} d\lambda\right]\left(-\boldsymbol{H}^{-1}\boldsymbol{\Omega}\right) \\
\boldsymbol{\Gamma}_{\boldsymbol{\rho}} &= \left[\int_{t_k}^{t_{k+1}} e^{-\boldsymbol{H}^{-1}\boldsymbol{C}(\boldsymbol{\nu}(t_k))\left(t_{t_k+1}\right)} d\lambda\right]\left(-\boldsymbol{H}^{-1}\boldsymbol{\Xi}\right) \\
\boldsymbol{\Gamma}_{\boldsymbol{M}} &= \left[\int_{t_k}^{t_{k+1}} e^{-\boldsymbol{H}^{-1}\boldsymbol{C}(\boldsymbol{\nu}(t_k))\left(t_{t_k+1}\right)} d\lambda\right]\left(\boldsymbol{H}^{-1}\right) \\
\boldsymbol{\Gamma}_{\boldsymbol{M}} &= \boldsymbol{\Gamma}_{\boldsymbol{G}} = \boldsymbol{\Gamma}_{\boldsymbol{fric}}
\end{aligned}
\tag{4}
$$

Las funciones con exponentes matriciales que aparecen en (4) fueron resueltas en el simulador utilizando expansión en series de Taylor y el teorema de Cayley-Hamilton (Franklin et al., 1990).

Para terminar este apartado se incluyen algunos parámetros destacados del VAT modelado en este trabajo (Figura 1). Centro de gravedad: $x_G = -0{,}11m$, $y_G = 0m$, $z_G = 0{,}12m$. Componentes del tensor de inercia: $I_x = 0{,}1598Kgm^2$, $I_y = 0{,}1887Kgm^2$, $I_z = 0{,}178Kgm^2$, $I_{xz} = 0{,}0417Kgm^2$, $I_{zx} = I_{xz}$. Masa del robot: $19{,}36Kg$.

3. Controladores de velocidad

Al diseñar los controladores de velocidad para el VAT deben tenerse en cuenta ciertas consideraciones, que pueden hacer razonable la utilización de métodos de ajuste no analíticos:

- (2) es una ecuación en espacio de estados no lineal, dado que el componente (r) del vector de estados se encuentra tambien en la matriz $\boldsymbol{C}$.
- Las fricciones incrementan el problema de no linealidad y las matrices $\boldsymbol{C}$ y $\boldsymbol{\Xi}$ en (2) son además variantes en el tiempo.
- A bajas velocidades, los errores por truncamiento en los codificadores ópticos de las ruedas empiezan a ser significativos.
- El ruido que habitualmente acompaña a los acelerómetros de bajo costo, utilizados como inclinómetros, puede provocar una información equivocada acerca de la inclinación real del VAT.

Presentamos a continuación la formulación de los controladores evaluados.

3.1. Controlador PID

Los controladores PID no son especialmente adecuados para sistemas no lineales variantes en el tiempo. Sin embargo, se ha confiado en la realización de un ajuste adecuado, por utilizarse para ello un algoritmo genético que, en definitiva, produce un efecto similar a la linearización de la planta alrededor del punto de trabajo seleccionado.

Se presenta en primer lugar la ecuación predictora del vector error de velocidad $\boldsymbol{e}(t)$, en la que se utilizan las matrices proporcional $\boldsymbol{K_p}$, integral $\boldsymbol{K_i}$ y derivativa $\boldsymbol{K_d}$, y donde T representa el periodo de muestreo.

$$
\boldsymbol{e}\left(t_{k+1}\right) = -\boldsymbol{K_p}\boldsymbol{e}\left(t_k\right) - \boldsymbol{K_d}\frac{\boldsymbol{e}\left(t_k\right) - \boldsymbol{e}\left(t_{k-1}\right)}{T} - \boldsymbol{K_i}T\sum_{n=1}^{k}\boldsymbol{e}\left(t_n\right) \tag{5}
$$

Para calcular el vector error de velocidad actual del VAT, se resta el vector de velocidad actual $\boldsymbol{\nu_R}$ del vector de velocidad deseada $\boldsymbol{\nu_D}$.

$$
\boldsymbol{e}\left(t_k\right) = \boldsymbol{\nu_D}\left(t_k\right) - \boldsymbol{\nu_R}\left(t_k\right) \tag{6}
$$

Si se resuelve (3) para el vector $\boldsymbol{\tau_M}$, entonces obtenemos (7).

$$
\begin{aligned}
\boldsymbol{\tau_M}\left(t_k\right) = \left[\boldsymbol{\Gamma_M}\left(t_k\right)\right]^{-1} \big[&\boldsymbol{\nu_R}\left(t_{k+1}\right) - \boldsymbol{\Phi}\left(t_k\right)\boldsymbol{\nu}\left(t_k\right) \\
&- \boldsymbol{\Gamma}_{\dot{\boldsymbol{\rho}}}\left(t_k\right)\dot{\boldsymbol{\rho}}\left(t_k\right) - \boldsymbol{\Gamma}_{\boldsymbol{\rho}}\left(t_k\right)\boldsymbol{\rho}\left(t_k\right) \\
&- \boldsymbol{\Gamma_G}\left(t_k\right)\boldsymbol{\tau_G}\left(t_k\right) - \boldsymbol{\Gamma_{fric}}\left(t_k\right)\left(-\boldsymbol{\tau_{fric}}\left(t_k\right)\right)\big]
\end{aligned}
\tag{7}
$$

Entonces, si (5) y (6) se sustituyen en (7) obtenemos (8).

$$
\begin{aligned}
\boldsymbol{\tau_M}\left(t_k\right) = &\left[\boldsymbol{\Gamma_M}\left(t_k\right)\right]^{-1} \\
&\left[\boldsymbol{\nu_D}\left(t_{k+1}\right) + \boldsymbol{K_p}\boldsymbol{e}\left(t_k\right) + \boldsymbol{K_d}\frac{\boldsymbol{e}\left(t_k\right) - \boldsymbol{e}\left(t_{k-1}\right)}{T}\right. \\
&\left. + \boldsymbol{K_i}T\sum_{n=1}^{k}\boldsymbol{e}\left(t_n\right) - \boldsymbol{\Phi}\left(t_k\right)\boldsymbol{\nu}\left(t_k\right)\right. \\
&- \boldsymbol{\Gamma}_{\dot{\boldsymbol{\rho}}}\left(t_k\right)\dot{\boldsymbol{\rho}}\left(t_k\right) - \boldsymbol{\Gamma}_{\boldsymbol{\rho}}\left(t_k\right)\boldsymbol{\rho}\left(t_k\right) \\
&\left. - \boldsymbol{\Gamma_G}\left(t_k\right)\boldsymbol{\tau_G}\left(t_k\right) - \boldsymbol{\Gamma_{fric}}\left(t_k\right)\left(-\boldsymbol{\tau_{fric}}\left(t_k\right)\right)\right]
\end{aligned}
\tag{8}
$$

Finalmente, los voltajes de alimentación para los motores se obtienen resolviendo la ecuación eléctrica y mecánica de los mismos (Hernandez y Canas, 2011) con el valor $\boldsymbol{\tau_M}$ obtenido en (8).

3.2. Controlador Borroso

Este controlador ha sido diseñado bajo una perspectiva descentralizada. Una vez conocida la referencia de velocidad a respetar por el VAT, el controlador determina cuál es la velocidad que deben mantener cada una de las ruedas motrices, estableciendo entonces dichas velocidades como consignas a dos controladores borrosos independientes, que respetan el modelo de Mamdani con una estructura proporcional integral (PI) (Burns, 2001).

El sistema de codificación (*fuzzify*) diseñado, utiliza cinco etiquetas para los valores de entrada y salida, que son: negativo grande (NG), negativo pequeño (NP), cero (C), positivo pequeño (PP), positivo grande (PG).

Cada una de las etiquetas se corresponde con una función triangular específicamente ubicada en el eje de abscisas (Figura 3) y como consecuencia de todo ello, se plantean 15 funciones de pertenencia para definir cada controlador borroso (5 para el

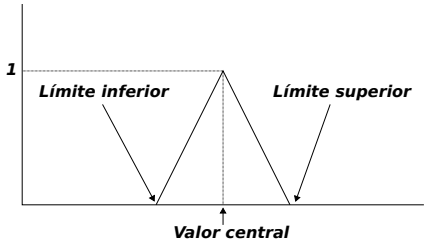

Figura 3: Función de pertenencia con formato triangular.

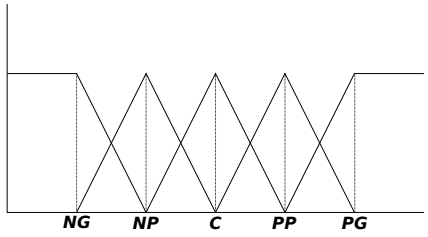

Figura 4: Patrón de funciones de pertenencia para señales de entrada y salida.

error, 5 para el sumatorio del error y 5 para la salida del controlador).

Las señales de entrada y salida se normalizan entre -1 y 1, por lo que el valor central de $NG = -1$ y el valor central de $PG = 1$. Al superponer las funciones de pertenencia se respeta el patrón planteado en la Figura 4.

Las reglas de inferencia para calcular las señales de salida se resumen en la Tabla 1, en la que las etiquetas horizontales se refieren al error actual (e) y las verticales al error acumulado ($\sum e$). Un ejemplo de interpretación de dicha tabla se recoge en la ecuación (9), en la que se presenta la forma de calcular la altura (C_{alt}) del triángulo truncado asociado a la etiqueta de salida C_o.

Tabla 1: Reglas del controlador borroso PI.

	$NG_e(e)$	$NP_e(e)$	$C_e(e)$	$PP_e(e)$	$PG_e(e)$
$NG_{\sum e}(\sum e)$	NG_{alt}	NG_{alt}	NP_{alt}	NP_{alt}	C_{alt}
$NP_{\sum e}(\sum e)$	NG_{alt}	NP_{alt}	NP_{alt}	C_{alt}	PP_{alt}
$C_{\sum e}(\sum e)$	NP_{alt}	NP_{alt}	C_{alt}	PP_{alt}	PP_{alt}
$PP_{\sum e}(\sum e)$	NP_{alt}	C_{alt}	PP_{alt}	PP_{alt}	PG_{alt}
$PG_{\sum e}(\sum e)$	C_{alt}	PP_{alt}	PP_{alt}	PG_{alt}	PG_{alt}

$$\begin{aligned} C_{alt} = &\left[PG_e(e) \wedge NG_{\sum e}\left(\sum e\right)\right] \vee \\ &\left[PP_e(e) \wedge NP_{\sum e}\left(\sum e\right)\right] \vee \left[C_e(e) \wedge C_{\sum e}\left(\sum e\right)\right] \vee \\ &\left[NP_e(e) \wedge PP_{\sum e}\left(\sum e\right)\right] \vee \left[NG_e(e) \wedge PG_{\sum e}\left(\sum e\right)\right] \end{aligned} \tag{9}$$

Se ha optado por utilizar la t-norma mínima (menor valor de los operadores de entrada) para la función lógica "and", representada por "∧". Para el caso de la operación lógica "or", representada por "∨" se ha utilizado la t-conorma máxima (el valor máximo de los operadores de entrada) (Passino y Yurkovich, 1998).

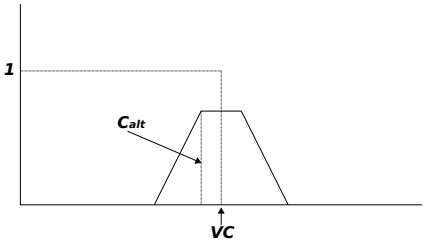

Figura 5: Ejemplo de triángulo truncado correspondiente a la función de pertenencia de la etiqueta de salida C_o, en donde C_{alt} se calcula según (9).

Finalmente, la salida de los controladores de tipo Mamdani se calcula de la siguiente manera:

1. Cada función de pertenencia de la salida (FP_o) se trunca en altura de acuerdo con las reglas de inferencia de la Tabla 1 como se refleja en la Figura 5.
2. La salida del controlador se calcula utilizando el método del centro de gravedad (10), siendo S_o la salida del controlador. FP_o puede tomar los valores NG_o, NP_o, C_o, PP_o o PG_o. La función $Area(FP_o)$ suministra el área del triángulo truncado correspondiente a una etiqueta de salida, utilizándose para ello también el límite inferior, el límite superior y el valor central ($VC(FP_o)$) de cada triángulo (Figura 3), valores que se fijan en el proceso de ajuste del controlador.

$$S_o = \frac{\sum_{FP_o} Area(FP_o)\, VC(FP_o)}{\sum_{FP_o} Area(FP_o)} \tag{10}$$

3.3. Controlador LQR adaptativo

Los reguladores del tipo LQR deben adaptarse en el caso de querer que resuelvan un problema de seguimiento de consigna, lo cual no implica una especial complicación. La mayor dificultad surge del hecho de ser una propuesta adecuada para sistemas lineales invariantes en el tiempo (Goodwin et al., 2001), circunstancias que no concurren habitualmente en el caso de los VAT.

En esta ocasión, además de contar con un ajuste por medio de un algoritmo genético, al igual que se propuso para el caso del controlador PID, existe una especial facilidad para plantear una variante adaptativa del controlador, que consiste en recalcular la configuración del controlador LQR (de horizonte finito), en cada ciclo del bucle de control, de forma que la planta del sistema se pueda considerar de parámetros constantes dentro de cada ciclo. Esta alternativa consume mucho tiempo en cálculos, pero es abordable si se considera la potencia actual de los microprocesadores.

De forma más concreta, considérese en primer lugar una función de coste convencional (11) para el controlador LQR. En ella, el vector $\boldsymbol{x}(k)$ representa la secuencia de valores que adopta el vector de estado del sistema, con índice k, $\boldsymbol{u}(k)$ es la secuencia de valores de salida del controlador, con índice k, $\boldsymbol{S}$

es la matriz de coste del estado final, $\boldsymbol{Q}$ es la matriz de coste del estado actual, $\boldsymbol{R}$ es la matriz de coste de la salida actual del controlador y por último N_e es el número de pasos impuestos para alcanzar la consigna establecida en el problema de control.

$$\begin{aligned} \boldsymbol{j} =& \frac{1}{2}\boldsymbol{x}^T(N_e)\,\boldsymbol{S}\boldsymbol{x}(N_e) \\ &+ \frac{1}{2}\sum_{k=0}^{N_e-1}\left[\boldsymbol{x}^T(k)\,\boldsymbol{Q}\boldsymbol{x}(k) + \boldsymbol{u}^T(k)\,\boldsymbol{R}\boldsymbol{u}(k)\right] \end{aligned} \tag{11}$$

Si consideramos que (12) es la ecuación de estados del sistema, entonces se puede obtener (13) aplicando el algoritmo de programación dinámica de Bellman (Albertos y Sala, 2004) a (11) y (12).

$$\boldsymbol{x}(k+1) = A\boldsymbol{x}(k) + B\boldsymbol{u}(k) \tag{12}$$

$$\begin{aligned} &\boldsymbol{u}(k-1) = -\left[\boldsymbol{R} + \boldsymbol{B}^T\boldsymbol{S}_k\boldsymbol{B}\right]^{-1}\boldsymbol{B}^T\boldsymbol{S}_k\boldsymbol{A}\boldsymbol{x}(k-1) \\ &\boldsymbol{u}(k-1) = -\boldsymbol{K}_{\boldsymbol{G}}(k-1)\boldsymbol{x}(k-1) \\ &\boldsymbol{S}_{k-1} = \boldsymbol{Q} + \boldsymbol{A}^T\boldsymbol{S}_k\boldsymbol{A} - \boldsymbol{A}^T\boldsymbol{S}_k\boldsymbol{B}\left[\boldsymbol{R} + \boldsymbol{B}^T\boldsymbol{S}_k\boldsymbol{B}\right]^{-1}\boldsymbol{B}^T\boldsymbol{S}_k\boldsymbol{A} \end{aligned} \tag{13}$$

La ecuación (13) suministra el procedimiento para calcular la secuencia de salida del controlador $\boldsymbol{u}(k-1)$, donde k toma valores desde N_e hasta 1.

Cada valor $\boldsymbol{u}(i)$ es función del valor desconocido del vector de estado $\boldsymbol{x}(i)$. Sin embargo, la secuencia puede resolverse hacia atrás cuando el algoritmo alcanza $\boldsymbol{u}(0)$, dado que el estado inicial $\boldsymbol{x}(0)$ se conoce.

Una configuración inicial aceptable para $\boldsymbol{S}_{N_e}$ es la matriz identidad y el estado final esperado debe asignarse a $\boldsymbol{x}(N_e)$.

Como ya ha sido comentado, este planteamiento resuelve el problema de regulación, pero el control de velocidad de un VAT es un problema de servo control. Para adaptar el planteamiento efectuado, se adopta la solución habitual de realizar el cambio de variable (14) que finalmente lleva a la salida del controlador planteada en (15).

$$\widetilde{\boldsymbol{x}}(k) = \boldsymbol{x}(k) - \boldsymbol{x}(k)_{deseado} \tag{14}$$

$$\widetilde{\boldsymbol{u}}(k) = -\boldsymbol{K}_{\boldsymbol{G}}(k)\widetilde{\boldsymbol{x}}(k) = -\boldsymbol{K}_{\boldsymbol{G}}(k)\left[\boldsymbol{x}(k) - \boldsymbol{x}(k)_{deseado}\right] \tag{15}$$

En cada ejecución del bucle de control se actualiza la ecuación de estado y lo mismo ocurre con el estado inicial y final. Por último, se selecciona como salida del controlador el primer elemento de la secuencia de salida.

4. Plan experimental

4.1. Interpretación estadística de los fallos del software

Si en un programa "sencillo" se introduce un defecto que provoca un fallo, dicho fallo habitualmente se produce de manera sistemática y no aleatoria. Sin embargo, la mayoría de los investigadores que trabajan en el estudio de la fiabilidad de los sistemas programados, asignan un comportamiento aleatorio a los fallos del software, en aquellos sistemas que tienen conjuntos de datos de entrada y salida muy grandes, junto con un elevado número de lugares en el código fuente en los que se pueden introducir defectos (Shooman, 2002). Si se piensa cuidadosamente, el clásico experimento estadístico de lanzar un dado y observar el resultado, admite una interpretación similar. Si conseguimos lanzar un dado un número elevado de veces, exactamente con las mismas condiciones, obtendremos el mismo resultado sistemáticamente. El problema es poder controlar eficazmente todas las condiciones que afectan al resultado cuando se lanza un dado. Esta dificultad justifica la interpretación estadística de los datos obtenidos en este experimento.

En los sistemas de control programados puede tener lugar una situación equivalente. Surge de la imposibilidad de verificar todos los escenarios posibles, lo cual puede hacer muy difícil detectar algunos defectos en el software. Aparece entonces el interés en la comparación de alternativas de control programadas, en relación con su fiabilidad. Dicho de otra forma, si un controlador sigue funcionando de una forma aceptable, desde un punto de vista estadístico, incluso con defectos en el software, cuando otros controladores fracasan, puede tener interés considerar esa mayor fiabilidad a la hora de seleccionar el controlador que finalmente se instale en un sistema. Por contra, si es necesario utilizar una alternativa de baja fiabilidad, el conocimiento de dicho comportamiento permite activar mecanismos de desarrollo del software más exigentes, que minimicen el riesgo de introducir defectos en el sistema.

4.2. Condiciones respetadas en los experimentos de fiabilidad

En el presente trabajo se plantearon un conjunto de restricciones que deberían ser respetadas por los diferentes controladores a comparar. Dichas limitaciones están relacionadas con la forma en que los controladores se comportan en régimen transitorio y en régimen permanente. El objetivo era obtener controladores, de diferente arquitectura (PID, LQR y borroso), ajustados para cumplir con las mismas condiciones de diseño y así poder comprobar, en igualdad de condiciones, si dichas clases de controladores ofrecen distintas fiabilidades cuando se ven afectados por algunos defectos habituales de programación.

Los recorridos serpenteantes son habituales cuando se realizan actividades de búsqueda, exploración, limpieza, etc. Por dicho motivo, en las pruebas que se han efectuado a los controladores evaluados, se ha introducido un componente de navegación que impone una trayectoria de este tipo. Dicho componente de navegación trabajaba en lazo abierto para hacer más evidentes los posibles fallos, en el control de velocidad, en que pudieran incurrir los controladores comparados. El componente de navegación emite la siguiente secuencia de órdenes dentro de un bucle que funciona durante 1250 s (tiempo del experimento).

- En primer lugar se impone una consigna de velocidad de la forma $u = 0{,}2\ m/s$ y $r = 0\ rad/s$ durante 100 s. Es decir, la consigna persigue que el VAT se desplace en línea recta con una velocidad de 0,2 m/s.

- En segundo lugar se cambia la consigna de velocidad a $u = 0{,}2\ m/s$ y $r = -0{,}1\ rad/s$ durante 31,416 s. En esta ocasión se pretende que el VAT recorra un arco de circunferencia a la derecha, para finalmente girar 180º.
- En tercer lugar se repite la consigna establecida en el primer tramo, es decir, $u = 0{,}2\ m/s$ y $r = 0\ rad/s$ durante 100 s. Con ello se persigue recorrer un segmento paralelo al del primer tramo, pero en sentido contrario debido el giro efectuado en el segundo paso.
- Para finalizar, se impone la consigna $u = 0{,}2\ m/s$ y $r = 0{,}1\ rad/s$ durante 31,416 s. Se pretende con ello que el VAT recorra un arco de circunferencia igual que el del segundo paso, pero en esta ocasión a la izquierda.

Para hacer ligeramente más complicado el control de velocidad, la prueba a los controladores se realiza sobre un suelo ondulado cuya altura respeta la ecuación (16), según sean las coordenadas del marco de referencia global x e y.

$$z = 0{,}25 \sin\left(\left(\frac{2\pi}{20}\right)x\right) + 0{,}25 \sin\left(\left(\frac{2\pi}{20}\right)y\right) + 0{,}5 \qquad (16)$$

En los estudios de fiabilidad es necesario determinar qué escenarios se consideran fallidos, por lo que en el presente trabajo se estableció una ventana de velocidades medias aceptables (media de 1000 muestras) para el estado estacionario y una ventana temporal para el estado transitorio. Si en algún ensayo la velocidad media del VAT cae fuera de la ventana permitida, la prueba del controlador en curso se aborta, anotándose el tiempo transcurrido hasta fallar.

La ventana máxima para el error medio en régimen permanente se estableció en $\pm 0{,}01 m/s$ para la velocidad longitudinal y $\pm 0{,}007 rad/s$ para la velocidad angular. Para el caso del régimen transitorio la ventana temporal establecida fue de $0{,}3s$ para la velocidad longitudinal y de $0{,}15s$ para la velocidad angular.

Dicho esto, definimos *controlador bien ajustado* como aquel que respeta las condiciones establecidas en esta sección.

4.3. Procedimiento para los experimentos de fiabilidad

Las investigaciones en fiabilidad del software frecuentemente se centran en el desarrollo de técnicas para medir la fiabilidad y en la generación de modelos de evolución de la fiabilidad. Con ello se persigue no poner en explotación sistemas poco fiables así como estimar el tiempo y esfuerzo necesario para alcanzar un determinado nivel de fiabilidad (Tian, 2005; Xie et al., 2004).

En la investigación que aquí se presenta se ha recuperado el planteamiento original de los estudios de fiabilidad, los cuales surgieron en paralelo con las técnicas de producción en cadena (Saleh y Marais, 2006; McLinn, 2011). En ellos habitualmente se seleccionaba una muestra del producto a desarrollar y, analizando el número de unidades defectuosas, se determinaba si una partida concreta tenía la calidad necesaria para ser comercializada.

En los experimentos realizados en el presente trabajo se han considerado disponibles 1000 programadores, los cuales debían generar, a partir de un mismo diseño sin defectos, el código fuente correspondiente a un controlador de velocidad. Los programas generados han sido sometidos a un ejercicio de fiabilidad según las condiciones establecidas en el apartado anterior (4.2). Con los tiempos hasta fallar obtenidos en las pruebas realizadas se han elaborado las curvas de fiabilidad por el método de *Kaplan-Meier*. Por desgracia, no hemos podido disponer de 1000 programadores, por lo que su trabajo se ha simulado utilizando programas sin defectos a partir de los cuales se han generado cientos de copias defectuosas por medio de un proceso de mutación.

De forma añadida y teniendo en cuenta el tiempo y esfuerzo que podría ahorrarse, en el plan experimental se planteo un subobjetivo previo, consistente en determinar si los controladores de un mismo tipo ofrecían la misma fiabilidad (desde un punto de vista estadístico) cuando eran sometidos al mismo patrón de defectos. De confirmarse este extremo, como de hecho ocurrió, se podría hacer un estudio de fiabilidad exhaustivo basado solamente en la evaluación de un representante de cada clase de controlador.

En el presente trabajo se ha considerado muy poco realista que un controlador se ponga en uso con más de cinco defectos sin detectar, por dicho motivo, cada uno de los controladores evaluados ha sido ensayado con un conjunto de defectos que varían desde uno hasta cinco.

Para comparar las fiabilidades obtenidas se ha utilizado el test *log-rank* (Kleinbaum y Klein, 2005). Los procedimientos *Kaplan-Meier* y *log-rank* son soportados por diferentes programas de análisis estadístico, como pueden ser *SPSS* (menú de análisis, opción supervivencia) o *R* (Fay y Shaw, 2010).

4.4. Simulador del VAT

En base a las directrices planteadas en los apartados anteriores, se decidió ajustar 30 controladores de cada tipo a evaluar (90 en total) que respetaran las restricciones planteadas en la sección 4.2.

De cada uno de los 90 controladores se decidió generar 1000 controladores mutantes, por lo que para obtener las curvas de fiabilidad se evaluaron 90000 controladores. Como cada una de las pruebas puede necesitar 1250 segundos, para verificar si existe la misma fiabilidad intraclase (estadísticamente hablando) son necesarios 3,57 años de operación sin paradas con un VAT. Más aún, durante el periodo experimental es necesario que el VAT no introduzca fallos de otra naturaleza que finalmente se puedan confundir con fallos generados por el software.

Debido a las dificultades anteriores, el equipo de investigación decidió construir un simulador multicomputador para llevar a cabo la etapa experimental.

En el simulador multicomputador, el código de los controladores fue modificado para permitir la inyección de defectos. Este planteamiento permitió la generación automática de un elevado número de clones defectuosos a partir de un controlador bien programado.

Los defectos en el software considerados en la investigación fueron los siguientes:

1. Defectos en números reales, los cuales consistieron en el cambio de un dígito o en el desplazamiento de la coma decimal (Jeffrey et al., 2008).
2. Defecto en operadores lógicos, que consistieron en la simulación del cambio de un operador en una expresión lógica de forma que el resultado final fuera el de la negación de la expresión original (Lau y Yu, 2005).
3. Defectos en operadores aritméticos, que consistieron en el intercambio de operadores en el caso de suma y resta, y en el caso de división y multiplicación (Gong et al., 2003).

El simulador multicomputador operó de la siguiente manera:

En primer lugar permitía elegir el número de defectos a inyectar (entre 1 y 5) en todos los clones a generar.

En segundo lugar, para cada uno de los defectos a introducir, se seleccionaba el tipo de defecto a generar. Para ello se tenía en cuenta el número de posibles puntos de inserción de cada tipo de defecto existente en el controlador con el que se estuviera trabajando. Para la elección concreta del tipo de defecto se utilizó un generador de números aleatorios que respetaba una distribución uniforme. Por ejemplo, si en el código de un controlador existen 250 puntos de inserción para defectos del tipo 1, 100 puntos para defectos del tipo 2 y finalmente 9 puntos para defectos del tipo 3, entonces la probabilidad de inyectar un defecto del tipo 3 era de 9/359.

En tercer lugar, después de elegir el tipo de defecto a inyectar, se seleccionaba el punto de inserción, para lo cual, se utilizó de nuevo el generador de numeros aleatorios anteriormente comentado.

En cuarto lugar, para el caso especial de tener que inyectar un defecto en un número real, se seleccionaba de nuevo, de forma aleatoria, si el defecto consistiría en el cambio de un dígito o en el desplazamiento de la coma decimal. Si se elegía modificar un dígito, se seleccionaba el mismo nuevamente de forma aleatoria. Si por el contrario se había optado por cambiar la posición del punto decimal, se determinaba aleatoriamente si adelantarlo o retrasarlo una posición.

En el presente proyecto, el simulador para ejecutar el algoritmo genético (AG) que debía obtener los 90 controladores bien ajustados (ver sección 4.2), es una adaptación del simulador para realizar las pruebas de fiabilidad.

4.5. Planificación de las pruebas de fiabilidad

El plan de pruebas para evaluar la fiabilidad de los controladores fue el siguiente:

1. Obtener 30 controladores bien ajustados de cada tipo de controlador (LQR, PID y borroso).
   Para este fin se propone utilizar un AG por los motivos indicados al principio del apartado 3, los cuales dificultan utilizar otras estrategias más habituales y por permitir este criterio someter a los controladores al mismo nivel de exigencia en el proceso de su ajuste.
2. Generar 1000 mutantes de cada uno de los 90 controladores anteriormente preparados, inyectando 5 defectos en cada uno de ellos.
   La razón por la que se propone inyectar cinco defectos es que las curvas de fiabilidad de cada tipo de controlador estarán más dispersas con un elevado número de defectos inyectados y cinco era el máximo que se había considerado utilizar.
   Si con cinco defectos las curvas de fiabilidad de un mismo controlador pueden considerarse iguales, es razonable suponer que lo mismo ocurrirá con menos defectos. Esta forma de proceder tiene el beneficio añadido de que el tiempo invertido en esta comprobación se reduce frente a otras alternativas, dado que a mayor número de defectos inyectados los controladores tardarán menos tiempo en fallar.
3. Probar los 90000 controladores mutantes para registrar sus tiempos hasta fallar. La prueba realizada con cada controlador y los márgenes de error admitidos deberán ser los detallados en el apartado 4.2.
4. Utilizando el procedimiento de Kaplan-Meier y los tiempos hasta fallar obtenidos en el apartado anterior, elaborar las 90 curvas de fiabilidad de los controladores ensayados.
5. Verificar con el test *log-rank* la hipótesis nula H0:"todos los controladores del mismo tipo tienen la misma fiabilidad". Para no rechazar la hipótesis, con un nivel de confianza del 95 %, es necesario que el *valor-p*[1] obtenido al realizar el test cumpla $p > 0{,}05$ (Good y Hardin, 2003; Montgomery y Runger, 2003).
6. Si la hipótesis anterior se confirma, entonces debe elegirse un representante de cada clase de controlador para efectuar una prueba de fiabilidad más exhaustiva y continuar por el paso 7. En caso contrario continuar por el paso 8.
7. Deben compararse los controladores por grupos según sea el número de defectos inyectados, debiendo variar los mismos entre un defecto y cinco. La hipótesis nula en esta nueva situación será H0:"los controladores de distinto tipo tienen la misma fiabilidad". Si no se cumpliera la hipótesis en todos los casos (uno a cinco defectos inyectados) y siempre fuera ventajoso el mismo tipo de controlador, se habría detectado el controlador software con más fiabilidad.
8. En el caso de no poder elegir en el paso 6 a un representante de cada tipo de controlador, habría que comprobar si la franja de mayor fiabilidad está significativamente dominada por los controladores de un mismo tipo, en todos los ensayos que pudieran realizarse inyectando defectos en el rango de uno a cinco. De no ser así, no podría declararse diferencia significativa en la fiabilidad de los controladores software comparados.

[1] Probabilidad de obtener un resultado más extremo que el alcanzado con la muestra analizada, considerando cierta la hipótesis nula.

5. Arquitectura del simulador multicomputador

Hoy en día es posible encontrar arquitecturas software, de muy diferentes tipos, adecuadas para un VAT. Sin embargo, la mayor parte se encuentran fuertemente influenciadas por trabajos pioneros en el ámbito de las arquitecturas deliberativas (Nilsson y Fikes, 1970; Albus, 1985; Albus et al., 1989), arquitecturas reactivas (Brooks, 1986) y arquitecturas híbridas (Arkin, 1987).

En el presente trabajo se ha asumido que el software del VAT está organizado por niveles (Bensalem et al., 2009; gon Roh et al., 2009) y concretamente el controlador de velocidad trabaja en el nivel más bajo, muy cerca del interfaz de los sensores y motores. El nivel de *navegación* se ubica inmediatamente encima del nivel de *control* y el nivel de *gobierno* encima del de *navegación*. En esencia, el nivel de *gobierno* debe determinar cómo se comporta el VAT a largo plazo. Cuando el nivel de *gobierno* decide que el VAT debe desplazarse, envía los correspondientes comando al nivel de *navegación*. El nivel de *navegación* entonces debe determinar la ruta a seguir entre la posición actual y la destino. Para terminar, el nivel de *navegación* envía consignas de velocidad al nivel de *control* para que el robot se desplace siguiendo la ruta establecida. El nivel de *control* lee los sensores y modifica los voltajes de alimentación de los motores para respetar las consignas de velocidad recibidas.

5.1. Simulador multicomputador para ajustar controladores

Como ya se ha comentado, se optó por utilizar un AG para ajustar los controladores, existiendo múltiples ejemplos de aproximaciones similares. Una realización coincidente en algunos aspectos con la aquí presentada es (Lewin y Parag, 2003), en la que los autores sintetizan un controlador MIMO descentralizado por medio de un AG.

En el presente trabajo, el AG operó en cooperación con un simulador del robot según la estructura planteada en la Figura 6.

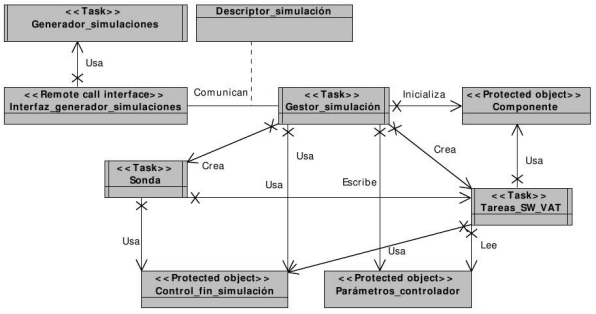

Figura 6: Componentes fundamentales del sistema multicomputador que ejecuta el AG de ajuste de controladores.

Uno de los computadores del "cluster" utilizado se reservó para la ejecución del módulo "Generador simulaciones" y su interfaz. Dicho componente contiene el núcleo del AG. Él produce la primera generación de controladores y elabora las siguientes a partir de los resultados de las evaluaciones que se realizan.

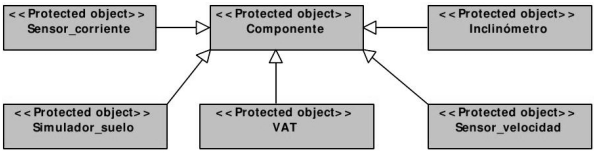

Figura 7: Componentes electromecánicos y simulador del suelo obtenidos por medio de herencia.

Figura 8: Componentes software del VAT obtenidos por medio de herencia

Los controladores se someten a la prueba planteada en la sección 4.2 en el módulo "Gestor simulación", existiendo cuatro unidades de dicho módulo en cada uno de los demás computadores (20 en total) del "cluster".

Concretamente, cuando el módulo "Generador simulaciones" desea evaluar un controlador, espera una petición de trabajo de un módulo "Gestor simulación", el cual recibe los nuevos parámetros de ajuste del controlador de velocidad a evaluar y produce un ensayo con el mismo.

Los componentes hardware del VAT y otros relacionados son refinamientos de la clase "Componentes" de la Figura 6, como puede verse en la figura 7.

Los componentes software del VAT son refinamientos de la clase "Tareas SW VAT" (Figura 8).

La sonda que aparece en la Figura 6, evalúa si la velocidad media que tiene el VAT se encuentra dentro de la ventana permitida. Si no es así, para la simulación y califica al controlador evaluado con el tiempo que ha tardado hasta fallar. Si se da la circunstancia de que dos controladores fallan en el mismo instante de tiempo, se tiene en consideración qué controlador ha tenido un error de velocidad máximo más pequeño. Para ello se registra la norma Euclídea del vector error velocidad en cada instante de muestreo de la sonda (cada 2 ms).

Cuando todos los controladores de una generación han sido evaluados, el algoritmo genético pone en marcha, de forma consecutiva, tres operadores para preparar la siguiente generación de controladores. Los operadores son *elitismo* (los mejores controladores de la generación actual pasan sin modificaciones a la siguiente generación); *combinación* (se generan nuevos controladores mezclando de forma aleatoria los componentes de dos controladores elegidos también de forma aleatoria); *mutación* (se genera un nuevo controlador cambiando de forma aleatoria alguno de los parámetros, de algunos controladores elegidos también de forma aleatoria).

En el presente trabajo el AG seleccionó por *elitismo* el 10 % de la nueva población, por *combinación* se generó el 60 %, (un 10 % se generó combinado aleatoriamente un miembro de la elite, elegido aleatoriamente, con otro elemento de la población elegido aleatoriamente y el 50 % restante combinando dos ele-

mentos elegidos aleatoriamente). El 30 % restante se eligió por *mutación* (un 10 % se correspondía a mutantes elegidos aleatoriamente de la elite de la población y el otro 20 % se obtenía mutando elementos de la población elegidos aleatoriamente).

Finalmente, para evitar controladores que generaran operaciones peligrosas (i.e. división por cero) o bucles infinitos, se diseño el componente "Control fin simulación", cuya misión principal era parar una simulación en dichas circunstancias.

El AG terminaba sus operaciones en el momento que conseguía un número preestablecido de controladores bien ajustados.

5.2. *Simulador multicomputador para los experimentos de fiabilidad*

Este simulador tiene una estructura similar al de la sección anterior (5.1), con la salvedad de que algunos módulos realizan actividades diferentes. Por ejemplo, en el código de los controladores a evaluar (LQR, PID y borroso) existen una serie de puntos en los que se pueden introducir defectos, de los tipos indicados en la sección 4.4. La tarea "Control" de cada "Gestor simulacion" ha sido modificada para permitir la activación condicional y configuración de los diferentes tipos de defectos que pueden tener lugar en ella. Del mismo modo, el módulo "Generador simulaciones" produce mapas de defectos y los envía a los módulos "Gestor simulacion" cuando estos reclaman trabajo. Estos últimos activan y configuran los defectos indicados en el mapa recibido, lanzando seguidamente una nueva simulación para determinar el comportamiento del controlador modificado. El módulo "Sonda" se encarga de determinar en qué momento falla cada controlador, registrando el instante en el que se produce el fallo. Dicho instante se devuelve al módulo "Generador simulaciones" el cual se encarga de preparar todos los datos para su posterior procesado estadístico.

5.3. *Simulación con tiempo virtual comprimido*

Cuando el módulo "Gestor simulación" arranca una nueva prueba, una visión más detallada de los componentes que se activan se ofrece en la Figura 9. Puede observarse que la tarea "Gobierno" (en este trabajo dicha tarea es un bucle infinito en el que únicamente se ejecuta una instrucción de espera pasiva cada varios segundos) envía órdenes a la tarea "Navegador" y esta hace lo mismo con la tarea "Control". La tarea "Control", como ya se ha indicado, recibe consignas de velocidad, siendo su misión conseguir el respeto de las mismas. Para ello consulta los sensores de velocidad y genera nuevos voltajes de alimentación a los motores, con la intención de mitigar el error de velocidad detectado.

El estado del VAT se encuentra encapsulado en el módulo "VAT", que en esencia es el simulador del vehículo. Dicho estado se actualiza bajo la demanda de la lectura de los sensores de velocidad, operación que siempre va acompañada de una marca de tiempo introducida por el módulo solicitante de la operación. Todos los controladores deben respetar un periodo de activación en su bucle principal de 1ms (necesario para tener un margen amplio en relación con el ancho de banda de los motores eléctricos modelados en el simulador), por lo que

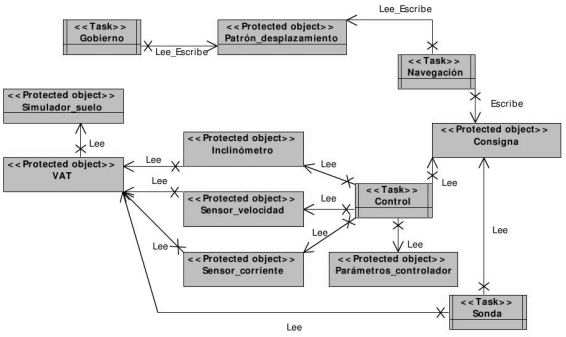

Figura 9: Perspectiva más detallada de los componentes de una simulación

el acceso a los sensores de velocidad se hace con una frecuencia altísima, lo que provoca que el estado del módulo "VAT" esté siempre correctamente actualizado.

Sin embargo, si los componentes software se ejecutan en un sistema real, existe la posibilidad de que en determinados intervalos de tiempo no exista ningún componente activo (Gobierno, Navegación y Control), es decir, todos estén bloqueados esperando su siguiente activación dentro de sus bucles principales. En los dos simuladores construidos para el presente trabajo (secciones 5.1 y 5.2), dichos intervalos de tiempo no se perdían, dado que el sistema operaba bajo un marco de tiempo virtual y cuando todas las tareas se encontraban en estado de espera, el tiempo virtual se avanzaba de forma artificial hasta la marca en que hubiera de despertarse a la primera tarea. El tiempo del simulador avanzaba en este caso más rápido que el tiempo fuera del simulador.

El simulador fue codificado en ADA95, utilizando GLADE como implementación del anexo E de computación distribuida (existe una implementación más reciente del anexo E, también de libre distribución, llamada POLYORB, pero este equipo de investigación todavía no ha migrado a ella).

Con la intención de evitar la recodificación del soporte de ejecución (*run time support*) para poder obtener las ventajas de operar con un marco de tiempo virtual, se implementó una solución basada en un objeto protegido con los servicios necesario para poder sustituir la instrucción "delay until" de ADA95. Dicho objeto protegido permitió a las tareas bloquearse hasta que el instante virtual declarado se alcanzara. El mismo objeto permitió detectar cuándo se encontraban bloqueados todos los procesos del sistema, lo cual es necesario para poder avanzar el reloj virtual como se ha comentado anteriormente.

Se seleccionó como política de planificación *programación cooperativa* (Burns y Wellings, 1997), que puede considerarse una variante del algoritmo de planificación *techo inmediato*. Se optó por dicha alternativa de planificación porque evita tener que realizar una verificación frecuente de cuál es el proceso más prioritario, lo cual permite efectuar simulaciones más rápidas por tener menos carga de trabajo el simulador.

A modo orientativo y de forma aproximada, por ser un parámetro de interés para la mayoría de los planificadores de tiempo real, parece interesante comentar el tiempo de cómputo requeri-

do, en cada ciclo, de los bucles principales de los controladores evaluados. Para obtener este resultado se ejecuto 1000 veces cada bucle de control de cada uno de los controladores analizados, utilizando en el experimento un procesador i5. En los controladores se eliminaron las instrucciones "delay until", así como las operaciones de lectura de sensores y las de escritura en los motores se sustituyeron por llamadas a procedimientos vacíos, dado que los retardos producidos por las mismas pueden considerarse despreciables. La duración de cada experimento (ejecución de 1000 ciclos) se registró utilizando el reloj del computador (se tomó el tiempo al principio del bucle de prueba y al final) dividiendo el resultado obtenido por 1000. Los datos obtenidos pueden considerarse, de forma aproximada, el tiempo invertido en la ejecución de su código por parte de cada uno de los controladores evaluados. En la tabla 2 se ofrecen los resultados, llamando la atención el poco tiempo de cómputo requerido por el controlador borroso en comparación con el LQR.

Tabla 2: Tiempo requerido por cada controlador para ejecutar el código en un ciclo de su bucle principal.

Controlador	*Tiempo consumido*
LQR	0,5 ms
PID	0,036 ms
borroso	0,005 ms

6. Resultados de los experimentos

Siguiendo la planificación de las pruebas de fiabilidad planteada en la sección 4.5, el simulador multicomputador, en su versión AG fue utilizado para obtener 30 controladores de cada uno de los tres tipos que se pretendía evaluar (30 LQR, 30 PID y 30 borroso).

En todos los casos, el AG fue configurado para obtener controladores que respetaran las ventanas de error establecidas en el apartado 4.2.

El simulador multicomputador, en su versión para experimentos de fiabilidad, se utilizó para realizar los pasos 2 y 3 del plan de pruebas. Por ello, se generaron 1000 mutantes defectuosos de cada controlador sin defectos (90000 en total). Cada uno de los controladores se alteró introduciendo cinco defectos elegidos aleatoriamente. Se efectuaron las pruebas de fiabilidad obteniéndose 90000 tiempos hasta fallar.

Como se estableció en el paso 5 del plan experimental, los tiempos obtenidos se utilizaron para generar las curvas de fiabilidad (Figura 10) de cada controlador (90 curvas en total) utilizando el procedimiento de Kaplan-Meier.

En el paso 5 del plan experimental se estableció la hipótesis nula H0:"todos los controladores del mismo tipo tienen la misma fiabilidad", la cual fue verificada por medio del test "log-rank", utilizando los datos experimentales obtenidos para obtener las curvas de fiabilidad. En todos los casos los valores "p" obtenidos (Tabla 3) no permiten rechazar la hipótesis nula (obsérvese que todos los valores-p son muy próximos a 1). Es

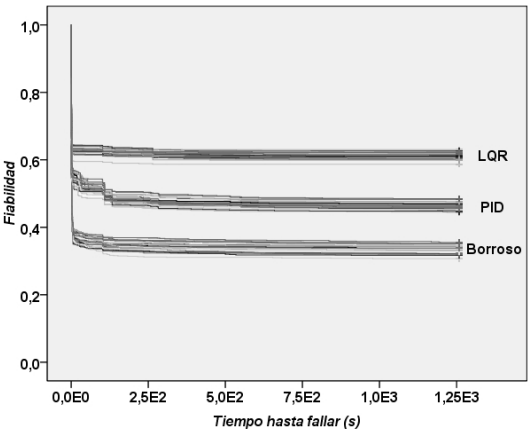

Figura 10: Curvas de fiabilidad de 30 controladores LQR (arriba), 30 controladores PID (en medio) y 30 controladores de tipo borroso (abajo).

decir, todos los controladores LQR, obtenidos según se ha indicado en la sección 5.1 tienen la misma fiabilidad (desde un punto de vista estadístico) y lo mismo ocurre con los controladores PID y borroso.

Tabla 3: Valores-p obtenidos al efectuar el test log-rank para contrastar que los controladores del mismo tipo tienen la misma fiabilidad.

Controlador	*p*
LQR	0,984
PID	0,975
borroso	0,927

Como la hipótesis nula se ha confirmado con cinco defectos inyectados, tiene sentido pensar que se obtendrán resultados menos dispersos al introducir menos defectos en los controladores, por lo que la hipótesis se mantendrá en dichas situaciones.

Atendiendo a lo establecido en el paso 6 del plan experimental y dado que los controladores de una misma clase ofrecen la misma fiabilidad (estadísticamente hablando), se seleccionó un controlador representante de cada clase. Con cada uno de dichos controladores se efectuaron cinco ensayos añadidos. En cada uno de los 5 ensayos efectuados con cada controlador se generaron 1000 mutantes, diferenciándose los mutantes de un ensayo con los de otros ensayos en el número de defectos inyectados, variando los mismos desde uno hasta cinco. Los 1000 tiempos hasta fallar obtenidos en cada ensayo se utilizaron para obtener las correspondientes curvas de fiabilidad, nuevamente utilizando el método de Kaplan-Meier (Figura 11 a Figura 15).

Por último, en el paso 7 del plan experimental se propone verificar la hipótsis nula H0:"los controladores de distinto tipo tienen la misma fiabilidad", pero ahora utilizando los datos obtenidos en el paso 6. En este caso la hipótesis nula fue rechazada en los cinco casos planteados, al haber obtenido en todas las situaciones un valor $p < 0{,}001$. Por ello, se puede afirmar que los tres controladores ensayados (LQR, PID y borroso) tienen

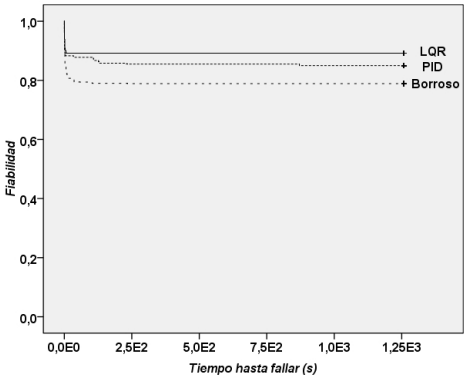

Figura 11: Curvas de fiabilidad de los representantes de los controladores LQR, PID y borroso con un defecto inyectado.

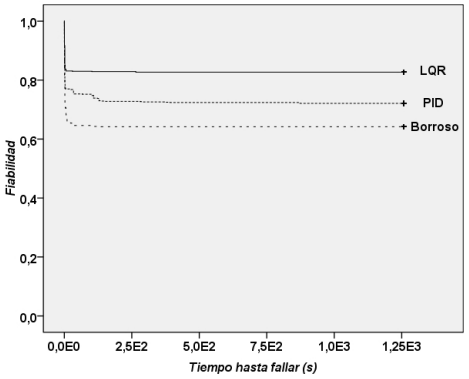

Figura 12: Curvas de fiabilidad de los representantes de los controladores LQR, PID y borroso con dos defectos inyectados.

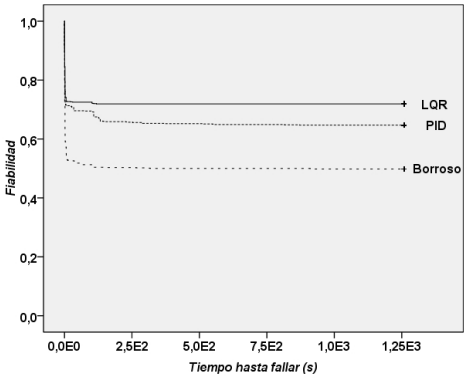

Figura 13: Curvas de fiabilidad de los representantes de los controladores LQR, PID y borroso con tres defectos inyectados.

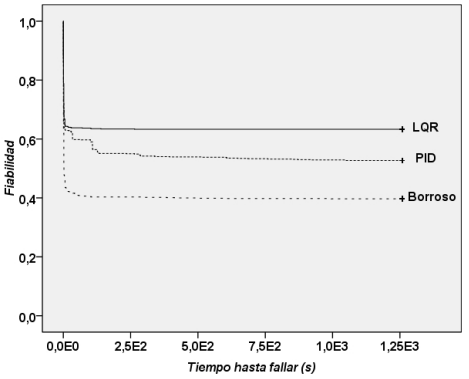

Figura 14: Curvas de fiabilidad de los representantes de los controladores LQR, PID y borroso con cuatro defectos inyectados.

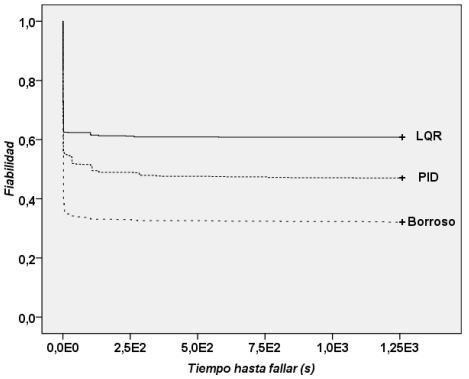

Figura 15: Curvas de fiabilidad de los representantes de los controladores LQR, PID y borroso con cinco defectos inyectados.

distinta fiabilidad y como se deduce de los valores registrados en las figuras 11 a 15, en todos los casos el controlador LQR es más fiable que el PID y el borroso, y el controlador PID es más fiable que el borroso.

7. Conclusiones

El diseño y construcción de un simulador específico, para poder ensayar el software de control embarcado en un VAT, ha permitido lograr el objetivo planteado. Se han comparado tres controladores de velocidad (LQR, PID y borroso) para un VAT, desde la perspectiva de la fiabilidad del software. En las condiciones experimentales planteadas, el controlador LQR es más fiable que el controlador PID y el controlador PID más fiable que el controlador borroso. No obstante, este último consume significativamente menos tiempo de CPU que las otras dos alternativas analizadas.

English Summary

Multivariable controllers for an autonomous ground vehicle: comparison based on software reliability

Abstract

In this paper, three multivariable speed controllers (linear quadratic regulator -LQR, proportional integral derivative - PID, and Fuzzy) were compared with each other to find which one has the best software reliability. The reliability tests were conducted on perturbed controllers with injected faults, simulating typical programmer errors. These controllers were designed to operate in an autonomous ground vehicle, and they were tuned by using a genetic algorithm. Given the large number of tests to be performed it was decided to build a multi-computer simulator in which they were carried out more than 90000 essays. In each of the trials, the perturbed controllers were subjected to a tour of approximately 20 minutes on a slightly wavy ground. With the obtained data, the reliability curves were elaborated by means of the Kaplan-Meier procedure, and this allowed their comparison which was the aim of this research. Under the observed experimental conditions, the LQR controller provides the best behavior, the second position belongs to the PID controller, and the third one to the fuzzy controller.

Keywords:

Software reliability Autonomous mobile robots Simulators LQR controller PID controller Fuzzy controller.

Agradecimientos

Esta investigación ha sido financiada por el Ministerio de Ciencia e Innovación (MICINN) de España bajo el proyecto de investigación TEC2010-17429.

Referencias

Albertos, P., Sala, A., 2004. Multivariable Control Systems: An Engineering Approach. Springer, pp. 303–309.

Albus, J. S., Aug. 1985. Hierarchical Control for Robots and Teleoperators. En: IEEE Workshop on Intelligent Control. pp. 1–14.

Albus, J. S., McCain, H. G., Lumia, R., 1989. NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM). NIST Technical note 1235. Tech. rep., National Institute of Standards and Technology.

Arena, P., Fortuna, L., Frasca, M., Turco, G. L., Patané, L., Russo, R., 2005. A new simulation tool for action-oriented perception systems. En: Proceedings of 10th IEEE Conference on Emerging Technologies and Factory Automation . pp. 571–577.

Arkin, R. C., Mar. 1987. Motor Schema Based Navigation for a Movile Robot. En: Proceedings of the IEEE International Conference on Robotics and Automation. pp. 264–271.

Bensalem, S., Gallien, M., Ingrand, F., Kahloul, I., Thanh-Hung, N., 2009. Designing Autonomous Robots. IEEE Robotics and Automation Magazine 16 (1), 67–77.

Brooks, R. A., Mar. 1986. A Robust Layered Control System for a Movile Robot. IEEE Journal of Robotics and Automation RA-2 (1), 14–23.

Burns, A., Wellings, A., 1997. Real-Time Systems and Programming Languages. Addison-Wesley, pp. 422–424.

Burns, R. S., 2001. Advanced Control Engineering. Butterwoth-Heinemann, pp. 326–347.

Carlson, J., Murphy, R. R., Jun. 2005. How UGVs Physically Fail in the Field. IEEE Transaction on Robotics 21 (3), 423–437.

Cheng, T. Y., Merkel, R., Jun. 2008. An Upper Bound on Software Testing Effectiveness. ACM Transcations on Software Engineering and Methodology 17 (3), 16:1–16:27.

Fay, M. P., Shaw, P. A., Aug. 2010. Exact and Asymptotic Weighted Logrank Test for Interval Censored Data: The Interval R Package. Journal of Statistical Software 36 (2), 1–34.

Fossen, T. I., 1999. Guidance and Control of Ocean Vehicles. John Wiley & Sons, pp. 5–28.

Franklin, G. F., Powell, J. D., Workman, M. L., 1990. Digital Control of Dynamic Systems. Addison Wesley, Ch. 2.

gon Roh, S., Yang, K. W., Park, J. H., Moon, H., Kim, H.-S., Lee, H., Choi, H. R., Apr. 2009. A Modularized Personal Robot DRP I: Design and Implementation. IEEE Transaction on Robotics 25 (2), 414–425.

Gong, Y., Xu, W., Li, X., Nov. 2003. An Expression's Single Fault Model and the Testing Methods. En: Proceedings of the 12th Asian Test Symposium (ATS'03) .

Good, P. I., Hardin, J. W., 2003. Common Errors in Statistics (and How to Avoid Them). John Wiley & Sons, Ch. 7, p. 100.

Goodwin, G. C., Graebe, S. F., Salgado, M. E., Jan. 2001. Control System Design. Prentice-Hall, pp. 672–674.

Henry, J., Stiff, J. C., Shirar, A. J., 2003. Assessing and Improving Testing of Real-time Software using Simulation. En: Proceedings of the 36th Annual Simulation Symposium. pp. 266–272.

Hernandez, W., Canas, N., Nov. 2011. Non-linear Control of an Autonomous Ground Vehicle. En: Proceedings of the 37th Annual Conference of the IEEE Industrial Electronics Society (IECON-2011). IEEE Industrial Electronicas Society, pp. 2601–2606.

Hydromechanics-Subcommittee, Apr. 1950. Nomenclature for Treating the Motion of a Submerged Body Through a Fluid. Tech. Rep. 1-5, The Society of Naval Architects and Marine Engineers (SNAME), New York (USA).

IEEE-Reliability-Society, Mar. 2008. IEEE Recommended Practice on Software Reliability. IEEE Std 1633-2008. Tech. rep., IEEE Reliability Society, 3 Park Avenue, New York, USA.

Isermann, R., 2008. Mechatronics systems-Innovative products with embedded control. Control Engineering Practice 16, 14–29.

Jeffrey, D., Gupta, N., Gupta, R., 2008. Fault localization using value replacement. En: Proceedings of the 2008 international symposium on Software testing and analysis (ISSTA '08). pp. 167–177.

Kleinbaum, D. G., Klein, M., 2005. Survival Analysis. A Self-Learning Text. Springer, Ch. 2.

Lau, M. F., Yu, Y. T., Jul. 2005. An Extended Fault Class Hierarchy for Specification-Based Testing . ACM Transactions on Software Engineering and Methodology (TOSEM) 14 (3), 247–276.

Lewin, D., Parag, A., 2003. A constrained genetic algorithm for decentralized control system structure selection and optimization. Automatica 39, 1801–1807.

McLinn, J., Jan. 2011. A short history of reliability. The Journal of the Reliability Information Analysis Center.

Montgomery, D. C., Runger, G. C., 2003. Applied Statistics and Probability for Engineers, 3rd Edición. John Wiley & Sons, Ch. 9.2.2.

Nilsson, N. J., Fikes, R. E., Oct. 1970. STRIPS: A New Approach to the Application ot Theorem Proving to Problem Solving. Technical note 43. SRI Project 8259. Tech. rep., Artificial Intelligence Group. Standford Research Institute.

Passino, K. M., Yurkovich, S., 1998. Fuzzy Control. Addison Wesley, pp. 58–73.

Raffo, G. V., Normey-Rico, J. E., Rubio, F. R., Kerlber, C. R., Jan. 2009. Control Predictivo en Cascada de un Vehículo Autónomo. Revista Iberoamericana de Automática e Informática Industrial 6 (1), 63–71.

Saleh, J., Marais, K., 2006. Highlights from the early (and pre-) history of reliability engineering. Reliability Engineering and System Safety 91, 249–256.

Sánchez-Peña, R. S., Sznaier, M., 1998. Robust Systems. Theory and Applications. John Wiley and Sons, Ch. 1.1.2.

Shooman, M. L., 2002. Reliability of Computer Systems and Networks. John Wiley & Sons, pp. 203–205.

Short, M., Pont, M. J., Fang, J., 2008. Assessment of performance and dependability in embedded control systems: Methodology and case study. Control Engineering Practice 16, 1293–1307.

Smidts, C., Huang, X., Widmaier, J. C., 2002. Producing reliable software: an experiment. The Journal of Systems and Software 61, 213–224.

Tian, J., 2005. Software Quality Engineering. Testing, Quality Assurance and Quantifiable Improvement. John Wiley and Sons, Ch. 22.1-22.4, pp. 371–380.

Xie, M., Dai, Y.-S., Poh, K.-L., 2004. Computing Systems Reliability. Models and Analysis. Kluwer Academic Publishers, Ch. 4.