# El uso de algoritmos en la introducción de los conceptos de variable y sucesión

Fecha de recepción: Enero, 1997

**Angel Marín**
Universidad Pública Navarra
Departamento de Matemáticas e Informática
amarin@unavarra.es

ARTÍCULOS DE INVESTIGACIÓN

*Educación Matemática*
*Vol. 12 No. 1 Abril 2000*
*pp. 97--109*

**Resumen.** *Este trabajo pretende destacar el interés educativo que podría tener un enfoque discreto en los métodos del cálculo infinitesimal, poniendo de relieve el uso que se hace de los conceptos de variable y sucesión en el desarrollo de algoritmos, así como el papel jugado en este contexto por los lenguajes de programación y por el propio lenguaje natural.*

**Abstract.** *This paper intends to underline the educational interest of a discrete approach of the calculation methods and the uses of the variable and succession concepts which are involved in the algorithm development process, also emphasizing the role played in this context by the natural and programming languages.*

## Lo continuo y lo discreto

A menudo la presentación de la teoría de funciones, con el consiguiente estudio de su continuidad, de los métodos de aproximación o de las condiciones de convergencia, hace olvidar, a quienes abordan el Cálculo con una visión marcada por el Análisis y la Topología, los procesos de cálculo efectivo que se concretan en algoritmos. En esa línea de presentación de los conceptos, los algoritmos cumplen, a lo sumo, un discreto papel como instrumentos para la verificación efectiva de las proposiciones establecidas mediante el análisis. Más tarde, una vez que han sido recalificados como métodos de cálculo numérico, son cedidos con alivio al dominio de los especialistas en computación.

Este tipo de enfoque, y los comportamientos a que da lugar, conforman el patrón educativo que habitualmente rige en la matemática universitaria, y que acaba heredándose en las matemáticas escolares, por un principio de subordinación metodológica casi siempre de dudosa oportunidad. Porque, al menos en este entorno escolar, debería discutirse si conviene o no dar prioridad a los métodos analíticos. En realidad, a poco que entremos en la cuestión, observaremos que lo que está de por medio es una forma de entender el infinito y los conceptos en que se sustenta.

Una primera idea de infinito, implícita en el recuento numérico, es la que sirve de base a la formación de iteraciones, constituyendo una abstracción de marcado carácter constructivo. Pero la formulación de un concepto tan básico para el Cálculo como el de número real, mediante las cortaduras de Dedekind, exige otro tipo de abstracción de mayor poder operativo en la que emerge la vieja concepción del infinito actual. En su

*Breve Historia del infinito* señala Zellini (1980) la identidad de criterio que liga en este sentido los proyectos de Bolzano, Cantor y Dedekind. A la hora de llevar estos conceptos al aula, debemos pensar en la asunción progresiva de esta abstracción, que sólo puede alcanzarse mediante los conceptos de sucesión y límite, y un sistemático ejercicio numérico asociado al infinito potencial.

Ambas versiones del infinito, que han sido entendidas de forma diversa a lo largo de los siglos, se remontan a Aristóteles, que distingue entre infinito actual o *enérgeia*, e infinito potencial o *dýnamis*. Con ellas no se pretende tanto ofrecer un criterio de magnitud como una forma de fijar lo indeterminado, que en el caso del infinito actual queda reflejado como un elemento simbólico y operativo, y en el del potencial como un proceso constructivo.

La exigencia de rigor, que a veces prima en la presentación de los conceptos de sucesión y límite, viene a conceder al infinito amplia operatividad en lo algebraico, mientras con ella se soslaya el desarrollo numérico efectivo y el carácter constructivo que es la base de su representación. En este sentido, la solución responde más al espíritu de Weierstrass que al de Euler. En términos de aprendizaje, la vía de acceso a la idea de límite y de continuidad que, siguiendo a Weierstrass, pasa por su expresión en términos de $\varepsilon$ y $\delta$-entornos, puede llegar a ser tortuosa, toda vez que exige una lenta maduración conceptual y no se ve apoyada y confirmada por sus posibilidades operativas numéricas (tan características para el alumno de la etapa educativa anterior, en la Aritmética), al omitirse su origen y representación en forma de algoritmos.

Este aspecto que hemos tomado como referencia de los métodos analíticos no es casual y apunta a uno de los grandes hitos del progreso matemático: la aritmetización del infinito.

Para dar relieve a este hecho, es necesario que nuestros estudiantes vean en el límite un modelo de representación e intervención numérica, y ésto sólo es posible si previamente adquiere significación como proceso de cálculo. En las presentaciones convencionales, basadas en entornos, no se dispone de medios para construir un modelo semejante. Sin el concurso de la Dinámica y sin el recurso a los algoritmos, presentes en las formulaciones iniciales de Newton y Leibniz, los esquemas formales que sirven de soporte a estos conceptos pueden resultar al alumno carentes de sentido, e incluso retóricos. Para evitarlo es preciso poner de relieve la relación que existe entre lo discreto y lo continuo, e ir mostrando los modelos algorítmicos como vías de integración de lo concreto en lo abstracto.

Será de poco provecho en la enseñanza secundaria hacer prevalecer un estilo educativo que tienda a la reproducción de conceptos consolidados y cerrados, y aplace su proyección en los problemas reales, o al menos en marcos en que el alumno ha desarrollado cierta destreza operativa. Tampoco tendría mucho sentido dar un papel preponderante a los algoritmos y a la Matemática discreta. Lo que ya no cabe es una discusión sobre su importancia, abierta en términos provocadores por A. Ralston (1984) con su artículo *Will Discrete Mathematics surpass Calculus in importance?*, si consideramos la amplia difusión de los actuales procesos de mecanización del Cálculo. En esta nueva etapa, caracterizada por el uso sistemático de los métodos finitos y por la necesidad de dar concreción, y forma numérica, a los métodos analíticos, los docentes en general y los diseñadores curriculares en particular deben de interrogarse sobre cuál sería la función educativa de estos métodos, cómo repercutiría su incorporación sobre el curriculum actual y en qué forma debería darse respuesta a su creciente demanda.

El fenómeno no parece coyuntural. Pretender que el auge de la computación, y con ella de la Matemática discreta, se ha producido a fin de dar cobertura a los nuevos instrumentos electrónicos de cálculo, es restringir su proyección al dominio numérico o matemático. Pero, en la actualidad, es innegable que la computación y el uso de algoritmos responden también a las necesidades de ciencias como la economía o la lingüística en las que se impone lo que Goguen (1968) llamó la lógica de lo inexacto. Por tanto, aunque como corriente matemática es de origen instrumental, la Matemática discreta ha dejado ver su calado teórico y, a través de sus aplicaciones a ciencias sociales, ha dado ocasión para reflexionar sobre las limitaciones de los métodos científicos basados en el continuo, propios del Cálculo infinitesimal.

## El empleo de algoritmos en el aprendizaje

El profesor de secundaria debería tener en cuenta esta situación cambiante respecto al papel de la Matemática discreta, y buscar un equilibrio que devuelva a los algoritmos su papel y su función en el marco de la formación matemática. La primera exigencia, en este sentido, consiste en evitar, tal y como indica Maurer (1991), que la idea de algoritmo quede exclusivamente asociada a los métodos y reglas propios del cálculo aritmético.

En todo caso el uso del algoritmo constituye un conocimiento procedimental, cuyo estudio es difícil de encuadrar como tema específico en una secuenciación de contenidos, y que encuentra mejor acomodo en el marco más general de la resolución de problemas. Esta ubicación evita que el algoritmo sea interpretado como mera mecánica o regla operativa, o como ilustración procedimental de contenidos pretendidamente teóricos. Desde ella, por el contrario, se puede recalcar el valor expresivo del algoritmo, y su importancia a la hora de favorecer la formación de nuevos conceptos mediante el desarrollo de las destrezas operativas, y también lingüísticas, que su uso comporta.

El interés y la viabilidad educativa del algoritmo tampoco están subordinados a la previa adquisición de un lenguaje algorítmico. En un nivel inicial, para instrumentar su uso, basta con la ayuda de los medios de expresión disponibles, es decir con el propio lenguaje natural. La llegada de los lenguajes de programación como instrumentos de especificación de procesos debe darse después, a menos que cedamos a la tentación de convertir su conocimiento en un factor de relieve social y los usemos para convertir el acceso de los jóvenes a los computadores en una especie de iniciación ritual. Este ejercicio de falsificación del aprendizaje de la algorítmica por el que se prescinde del lenguaje natural como instrumento descriptivo elemental a cambio de entrenar al alumno en un juego simbólico, atractivo en su economía y eficaz en su capacidad combinatoria, pero sin referencia ni historia, disminuye su capacidad para percibir el sentido de las cosas. Ciertamente los problemas no les serán comunicados en jerga o lenguaje artificial, ni su uso prematuro contribuirá a una interpretación adecuada de las situaciones problemáticas. Para evitar el deslumbramiento, y ese punto de fetichismo que convierte la programación en un intercambio de mensajes entre iniciados, es preciso hacer al alumno consciente de que la primera especificación de un proceso es la verbal, y que la expresión en lenguaje natural de una idea es un logro que permite concebir, integrar y dar una incipiente estructura lógica a experiencias dispersas. Servirse de este punto de apoyo es el mejor modo de dar solidez a la construcción de algoritmos, porque difícilmente podremos especificar y resolver un problema que somos incapaces de expresar.

Estas reflexiones nos llevan a considerar el empleo de algoritmos, en el ámbito de la resolución de problemas, como *una actividad que permite inicialmente encauzar la expresión verbal de las soluciones y posteriormente formular dichas soluciones mediante un proceso constructivo de cálculo.*

Al analizar estas dos facetas del uso algorítmico, descubrimos en la primera una función semejante a la reservada en la escuela al álgebra elemental, con la traducción o representación notacional de situaciones problemáticas; mientras que en la segunda, relativa a la construcción de procesos de cálculo, se adivina la importancia de contar con esquemas sintácticos (o de destacar aquellos propios del lenguaje natural) que favorezcan la expresión de secuencias operativas.

Las dos facetas consideradas apuntan, por tanto, a conceptos matemáticos fundamentales, entre los que destacan el de variable y el de sucesión. De ahí que se pueda atribuir al empleo de algoritmos en el marco de la resolución de problemas, tanto una función propedéutica como unificadora respecto al resto del material curricular. Propedéutica, puesto que la destreza adquirida con el uso de variables y sucesiones en el desarrollo de algoritmos daría referencias concretas y un sólido punto de apoyo a formulaciones más abstractas, como las que se dan en el Algebra. De ahí que su inclusión en el curriculum sirva para favorecer la incipiente formación de conceptos, que más tarde se revelarán fundamentales en el conocimiento matemático. Este mismo argumento hace ver cómo por medio de la familiarización con conceptos, tales como la variable y la sucesión, se convierte al algoritmo en un factor unificador de los recursos formales en que posteriormente se basarán el Algebra y el Cálculo.

## Progreso simbólico e idea de variable

Cualquier secuenciación de contenidos, destinada a elaborar un ciclo educativo, plantea abiertamente la necesidad de distinguir y elegir lo básico para dar con ello curso a lo complejo. Pero en ocasiones, más que esta elección temática importa conocer lo que los distintos métodos y disciplinas pueden aportar a la construcción de un concepto básico. Porque algunos de estos conceptos, surgidos en el seno de una determinada teoría, enlazan con criterios o esquemas de más alto rango epistemológico y son por tanto accesibles desde muy diversas metodologías científicas. Identificar este tipo de esquemas de alcance múltiple, investigar sus distintas vías de acceso y contribuir a su afianzamiento con métodos diversos son tareas que acaban teniendo una importante repercusión didáctica.

Se enfrentan en este punto dos concepciones del aprendizaje, una caracterizada por un ajuste lineal de la promoción cognitiva basado en el patrón temporal, y una segunda en que se crea el tejido cognitivo en torno a una serie de conceptos básicos, cuyo número y calidad de enlace se enriquece al encontrar para el conocimiento nuevas funciones y actividades.

De esta condición de básicos o fundamentales participan los conceptos ya mencionados de variable y sucesión, por lo que nos proponemos llevar adelante un análisis más detenido de la relación que a través de ellos se puede establecer entre distintas metodologías, y a la postre entre dos formas de ver las Matemáticas.

Teniendo en cuenta la importancia que estos conceptos tienen para la formación de los alumnos de secundaria, y la posibilidad de introducir ambos, tanto con métodos discretos como con métodos convencionales, podemos preguntarnos si existen aspectos específicos en los algoritmos, entendidos como métodos discretos, que hagan prevalecer

las ventajas de su empleo frente al uso tradicional de los métodos algebraicos y de cálculo infinitesimal.

En la primera de las facetas apuntadas anteriormente sobre el uso de algoritmos, indicábamos que con ellos se promueve la expresión de la solución mediante una especificación verbal, la cual puede dar paso más adelante a una traducción notacional. Ya sabemos que no es ésta la única vía, y que el mismo objetivo puede ser logrado a la vez por métodos puramente algebraicos como la formulación de ecuaciones y por métodos algorítmicos al describir secuencias operativas. Pero lo destacable es que en ambos casos se promueven estrategias heurísticas de propósito general para la resolución de problemas. De entre ellas quizá sea la más importante la elección de correspondencias entre la situación problemática y el marco de resolución. Por eso no debe extrañar que uno de los objetivos fundamentales en cualquiera de las dos perspectivas, algebraica o algorítmica, sea el fomento de la capacidad para identificar variables, y con ellas establecer esas correspondencias.

En algunos estudios realizados sobre la asunción por el alumno de los métodos algebraicos (Kücheman 1981), se presenta una gama escalonada de interpretaciones a través de las cuales el alumno va ampliando su concepto de variable, y no parece que en ambas perspectivas, algebraica y algorítmica, subyazca la misma gama de interpretación. En el enfoque algebraico, tal y como se recoge en dicho trabajo, se suele hacer progresar a la variable de la condición de *incógnita* lingüística a la de símbolo *neutro* caracterizado por su capacidad operativa. Este progreso, que podemos llamar progreso simbólico, representa la distancia que media entre un elemento de la notación algebraica y un nombre que encuentra su interpretación en el lenguaje natural. A medida que se significa en el símbolo su operatividad, disminuye el alcance de su denotación.

En el caso algorítmico la diversidad de interpretaciones sobre la variable no está determinada por el progreso simbólico, sino por el paradigma de programación que rige en la búsqueda de las soluciones a los problemas. En los tres paradigmas reconocidos: imperativo, funcional y lógico, el estatuto de la variable está asociado a la interpretación y función de los programas.

En el caso de los lenguajes imperativos, en los que los programas están constituidos por órdenes o instrucciones, la variable aparece como una etiqueta en el marco del sistema de almacenamiento, como el nombre de una caja, cuyo contenido y operaciones han sido previamente tipificadas. Sin embargo, en el paradigma lógico el programa es un conjunto de predicados sobre el que se formula una conjetura o hipótesis, lo que que obliga a la *instanciación*, o toma de valor, de las variables a fin de establecer sus condiciones de satisfacción. El uso de estas variables se corresponde con el habitual en las variables lógicas, es decir son símbolos cuya toma de valor hace cierto o falso el predicado. Por último, en el paradigma funcional el programa se describe como una composición de funciones, de modo que las variables cumplen papeles semejantes a los que asumen en el cálculo.

El criterio de progreso simbólico, con la correspondiente gama de interpretación que lo refleja, puede ser aplicado al estatuto propio de las variables en cada uno de los tipos de lenguaje, a fin de establecer algún tipo de correlación entre las diversas visiones de la variable aportadas por el Algebra y el Cálculo, y las propias de cada uno de los lenguajes de programación. Con esta aplicación se pretende describir con qué idea de variable característica del álgebra o del Cálculo encaja mejor la variable que se maneja en cada uno de los lenguajes. Esta tarea debe ir precedida de un análisis epistemológico de las ideas de variable que conforman lo que hemos denominado una gama sobre el

progreso simbólico, uno de cuyos modelos, recogido en el trabajo de Küchemann, venimos tomando como referencia.

A falta de resultados estadísticos y trabajos de campo, con ayuda de estas correlaciones presentaremos algunas conjeturas razonables y significativas, que servirán como orientación para fijar las exigencias que es preciso imponer a un lenguaje de programación para que el alumno logre en él una transcripción expresiva, y operativa con las variables, de la especificación verbal de la solución a su problema. Conclusiones de este orden pueden evitar la aparición o la persistencia de usos inconsistentes o contradictorios en las variables de los algoritmos y del álgebra, y pueden ser aplicadas, como veremos en el apartado siguiente, para representar de forma algorítmica y operativa ciertos conceptos del cálculo.

Para hacer la elección idónea es de gran interés establecer, a través de la idea de variable y de su prolongación en la idea de función, las correlaciones apuntadas y señalar los casos en que la cooperación entre álgebra y lenguaje de programación es más prometedora. Un repaso a las características de los lenguajes más comunes puede poner de manifiesto en qué casos se alcanza una mejor adecuación de las interpretaciones de variable propias del álgebra y el uso informático de las mismas.

Hay que tener en cuenta que, por lo común, la convivencia de ambos tipos de variable, algorítmica y algebraica, suele darse cuando el empleo de algoritmos llega a una aplicación en la que ya se ha utilizado el álgebra. Puede ser el caso del alumno que transcribe con un algoritmo las soluciones a una ecuación de segundo grado. Es preciso, entonces, dotar a la interpretación de variable que es más común en el álgebra, en concreto al caso en que se trata como incógnita o como signo sustituible, con una idea de almacenamiento propia de los lenguajes imperativos. El tránsito no parece severo, sobre todo si en el lenguaje elegido para el algoritmo prima, como en {\sc BASIC} por ejemplo, lo que Hoare (Hoare 1969) tilda de paradigma de la implementación automática de notaciones. La ventaja de este enfoque es que aprovecha íntegramente y refuerza el uso competente de la notación y convenios algebraicos. Pero el resultado de su aplicación restringe drásticamente los recursos lingüísticos, porque en tal paradigma la expresividad del algoritmo acaba donde termina la notación algebraica. No parece tampoco posible que dicho paradigma promueva el progreso simbólico, y llegue a favorecer, por ejemplo, la comprensión del uso de las variables en interpretaciones más complejas como puede ser la relación de correspondencia funcional. De modo que el encaje entre ambas ideas, al tiempo que favorece cierto reforzamiento del convenciones notacionales, puede estancar el progreso simbólico como tal.

Otro aspecto que puede inducir a confusión es la falsa concordancia entre las variables entendidas como identificadores, es decir estructuras nominativas características de estos lenguajes imperativos, y las notaciones algebraicas para variables y funciones. Se añade a esto, en el caso particular del {\sc BASIC} elemental, la falta de flexibilidad en la representación de funciones, la carencia de una estructuración de los datos y la imposible organización del código en estructuras sintácticas, lo que hace de este lenguaje un recurso dócil para prolongar cierta capacitación algebraica, pero limitado como vehículo de construcción de algoritmos.

En el caso del lenguaje LOGO la cuestión que nos ocupa ha sido tratada por algunos autores (Noss 1986) (Sutherland 1988). Ciertamente en este lenguaje se reúnen una serie de características que favorecen y potencian las estrategias constructivas de resolución. Destacan entre ellas su estructura procedimental, la organización funcional de los procedimientos y la incorporación de variables locales semejantes a las de los

lenguajes imperativos. Una revisión detenida de estos aspectos nos apunta algunas de las ventajas del lenguaje. En la situación, que parece más común, de incorporación del algoritmo al dominio establecido por el álgebra, este lenguaje parece más indicado, dado que incorpora y hace compatibles dos usos de la variable, el local y el global. El empleo de variables locales hace que las expresiones operativas sean similares a las algebraicas y el paradigma funcional lo hace, a su vez, particularmente indicado para representar en algoritmos cuestiones relacionadas con la naturaleza funcional del Cálculo. Además, por medio de la estructura procedimental se adaptan los enunciados y otros recursos lingüísticos al análisis de subproblemas y se favorece el desarrollo de estrategias heurísticas analíticas.

Podemos decir que este modelo de lenguaje genera nuevos usos de las variables que afianzan el concepto de función. Sin embargo, obliga a un uso recurrente de unos esquemas lingüísticos muy flexibles, pero sintácticamente pobres y alejados del lenguaje natural. Como instrumento en la representación de comportamientos funcionales y en la simulación de procesos propios de las ciencias experimentales (pensemos en fórmulas de la cinemática o modelos de crecimiento) puede llegar a ser idóneo. Pero no es difícil imaginar la complejidad de la representación en otro tipo de situaciones y problemas más abiertos. Su estructura, que no favorece la formación de secuencias que combinen acciones y operaciones, tampoco permite un adecuado aprovechamiento de la especificación verbal. Este distanciamiento del lenguaje natural acaba imponiendo un salto cualitativo al marcar con una diglosia (natural/artificial) la competencia algorítmica del alumno. En esta circunstancia, no siempre enmendable, se sacrifica el primero, el natural, y se delimitan con rigidez, y quizá demasiado pronto, los ámbitos de los dos tipos de lenguaje.

La pretensión del lenguaje {\sc PASCAL} es más modesta y fruto de una progresiva decantación o compromiso entre la descripción notacional y la sintaxis de los lenguajes naturales. La prioridad de la estructura secuencial lleva a un segundo plano la organización modular y la idea de función. En este lenguaje la descripción de procesos y operaciones se lleva a cabo de forma natural y la distancia que media entre la especificación verbal y el código es en muchos casos mínima. Por otro lado, la idea de variable asociada a la asignación de valores en memoria es perfectamente manejable si se dispone de formación algebraica. La estructura procedimental, aunque presente, ocupa un papel subordinado y, al menos en la etapa inicial, no hace competir en un mismo programa dos formas de ver la variable (como parámetro de paso y como identificación del almacenamiento) que pueden resultar contradictorias.

Finalmente, la variable lógica, y el paradigma que representa, es difícil de asumir en este nivel por el alumno, porque combina la naturaleza predicativa de la hipótesis con la instanciación numérica, y porque en los lenguajes más comunes, derivados de {\sc PROLOG}, este mecanismo de instanciación automática no es transparente al programador. Su empleo, en combinación con los métodos algebraicos y analíticos, requiere un notable grado de progreso simbólico e incluso cierto conocimiento de los métodos de demostración automática.

El balance ofrecido en este apartado, aunque modesto, apunta la posibilidad de analizar de forma más extensa y pormenorizada la correlación existente entre una gama de interpretación del progreso simbólico de origen epistemológico, tal como la de Kücheman, basada en el aprendizaje de las destrezas algebraicas, y los distintos estatutos fijados a las variables a través de la sintaxis y de la semántica de los distintos lenguajes de programación.

## Sucesión y principio de inducción

Una vez establecido el modo en que la primera de las facetas sobre el empleo de los algoritmos arriba anotadas apunta a una revisión del concepto de variable, recordemos que la segunda de dichas facetas ponía de relieve la utilidad de los algoritmos para formular la solución al problema mediante un proceso constructivo de cálculo.

La idea de procesos constructivos de cálculo nos remite a la de secuencia operativa. Aunque el orden de las operaciones es una cuestión importante en el álgebra, apenas se reconsidera al evaluar las expresiones algebraicas. Más bien se entiende la secuencia operativa, o calculograma, como una declaración de intenciones algorítmicas ajena a las transformaciones algebraicas. Ligado a ella surgen los procesos iterativos, cuyo fundamento matemático es el concepto de sucesión. Nos encontramos aquí, pues, no sólo ante un nuevo concepto fundamental para el posterior desarrollo del Cálculo Infinitesimal, y origen de las dos visiones del infinito comentadas anteriormente, sino ante el concepto mediador por excelencia entre lo continuo y lo discreto.

La existencia de estos dos enfoques sugiere también una doble vía, analítica y algorítmica, para afrontar la formación de dicho concepto. Una primera estrategia, que es la más comúnmente usada, pasa por destacar la relación del concepto de sucesión con el de función y su entronque con el resto del Cálculo. Se completa, más tarde, esta estrategia con un análisis del cálculo efectivo, que acaba por revelar su naturaleza inductiva y lo enlaza con las construcciones sintácticas en que se basan los algoritmos iterativos. Antes de presentar la segunda vía analicemos ésta con mayor detalle.

Para el matemático la importancia de las sucesiones está fuera de toda duda, ya que juegan un papel fundamental en el proceso de aritmetización del cálculo infinitesimal. Gracias a ellas se pueden expresar en forma discreta aproximaciones a los procesos que discurren o se establecen entre magnitudes basadas en el continuo real. El estudio de estos modelos de aproximación, enmarcado en la teoría de funciones, ha situado al concepto de función como colofón de todo el aprendizaje en la matemática escolar. Y este es el enfoque que orienta habitualmente la presentación de las sucesiones.

Un planteamiento útil en dicha presentación consiste en destacar en la sucesión la regularidad de sus términos. La idea de regularidad se asocia a la de ley de formación de los nuevos términos, y si los conocimientos algebraicos lo permiten, se plantea una reducción algebraica de la cuestión dando una expresión funcional al término genérico.

Este tipo de situación podría ilustrarse, por ejemplo, presentando los primeros términos en una sucesión

$$7, 14, 23, 34, \ldots$$

Al invitar al alumno a un estudio de la regularidad, se proponen diversas alternativas, que una vez contrastadas dan como siguientes los términos "47,62,79". Se le pide a continuación que dé forma algebraica a esa ley de formación que se intuye en los valores inferidos, y se concluye que el término *n*-ésimo sería

$$(n+2)^2 - 2$$

El planteamiento apunta ciertas virtudes: por un lado, se reconoce con una ley algebraica general la regularidad en la pauta de formación; pero, además, este resultado comporta la reversibilidad del proceso. De hecho, la síntesis algebraica lograda en el término general permite la construcción de los infinitos términos de la sucesión, con lo

que queda puesta de manifiesto la potencia expresiva de esa fórmula. El carácter sintético de la expresión, que engloba la totalidad de la sucesión, la convierte en instrumento idóneo para el desarrollo posterior de nuevos conceptos como el de convergencia, continuidad, límite de función, etc. Lo que viene a dar un indudable interés a este enfoque en la introducción del concepto de sucesión.

Situados en la segunda vía, parece difícil, en un principio, lograr con ayuda de algoritmos una síntesis que mejore la expresividad del término genérico o funcional.

Pero algunos de los problemas del enfoque analítico salen a la luz al tratar de emplear la expresión convenida, i.e.} el término genérico, en el proceso de cálculo.

Sin duda este elemento es paso obligado para la transcripción de la sucesión en el lenguaje de programación. Pero con la elección de esa fórmula como base de la construcción del algoritmo, hacemos que el proceso de cálculo se desarrolle fundamentalmente en clave algebraica. A efectos algorítmicos el cálculo se basa en la sucesiva toma de valor de una variable "n" en la fórmula, e indirectamente en la idea de variable *sustituible*. Así vemos que los términos se van obteniendo al sustituir "n" por 1, 2, 3,...

$$a_1 = (1 + 2)^2 - 2 = 7$$
$$a_2 = (2 + 2)^2 - 2 = 14$$
$$a_3 = (3 + 2)^2 - 2 = 23$$
$$\ldots$$

El empleo de esta interpretación de variable, nos sitúa en una de las primeras variantes en la gama de progreso simbólico. De modo que no parece forzar ninguna mejora en ese concepto, sino que se queda en una visión de la variable que no va más allá de la obtenida en el álgebra elemental. Pero, además, el hecho de basar el proceso en la fórmula resulta bastante pobre, e ineficiente, como recurso efectivo de cálculo.

El proceso de construcción de términos, así concebido, no resulta global sino *local*, en el sentido de que la constitución de un término se efectúa con independencia de los anteriores. Por ejemplo, para la obtención, a partir de la fórmula

$$a_n = (n + 2)^2 - 2$$

del término $a_{20}$ no tiene mayor interés el conocimiento de $a_{19}$ o los anteriores, aunque quizá este conocimiento aliviaría significativamente su cálculo. De hecho, se procede sustituyendo simplemente $n$ por 20. Cuando no se pretende un cálculo aislado sino que se plantea por ejemplo un proceso de aproximación, o cualquier otro que involucre a la sucesión entera, la ineficiencia se torna manifiesta. En términos de programación éste es un punto que debe ser considerado cuando se trata de optimizar el rendimiento de las máquinas.

Un ejemplo que muestra lo inconveniente de esta situación nos lo ofrece el cálculo aproximado del número "e". Una de las series que converge a "e" y que permitiría calcularlo con aproximación es

$$\sum_{n-1}^{\infty} \frac{1}{(n-1)!}$$

Si nos proponemos obtener una aproximación mediante la suma de 20 términos, y el término general se denota *an*, tendríamos

$$e = 1+1+\frac{1}{2!}+\frac{1}{3!}+\ldots+\frac{1}{19!}$$
$$= a_1 + a_2 + a_3 + a_4 + \ldots + a_{20}$$

Un cálculo posible nos llevaría a la aplicación reiterada de la fórmula que refleja el término general de la sucesión

$$a_n = \frac{1}{(n-1)!}$$

20 veces. Pero este enfoque, que calificábamos de local, mejora si consideramos que, una vez obtenido "$a_1$=1", se tienen las relaciones siguientes

$$a_2 = \frac{a_1}{1} \quad a_3 = \frac{a_2}{2} \quad a_4 = \frac{a_3}{3} \ldots a_{20} = \frac{a_{19}}{19}$$

En este nuevo enfoque que llamamos *global*, y que se basa en la construcción recursiva de la sucesión, cada término se obtiene a partir del anterior por simple división, evitando cálculos innecesarios. Además, la expresión general de la sucesión se adecúa más al algoritmo al ofrecerse las relaciones de recurrencia

$$\begin{cases} a_1 & = 1 \\ a_{i+1} & = \dfrac{a_i}{i} \quad 1 \le i < 20 \end{cases}$$

Esta representación puede extenderse, en forma recurrente también, a la sucesión de sumas parciales, cuyo término 20 define la aproximación del número *e* que buscamos.

$$\begin{cases} s_1 & = a_1 \\ s_{i+1} & = s_1 + a_{i+1} \quad 1 \le i < 20 \end{cases}$$

Así que, desde el punto de vista computacional, la expresión global de la sucesión mediante relaciones de recurrencia parece más indicada para el uso algorítmico que la que con carácter local se hace a base del término general.

Provisionalmente, el estudio desarrollado en torno al ejemplo anterior, que podría extenderse a otros muchos casos, muestra dos cosas: la primera, que desde una visión computacional de las matemáticas es posible operar sobre contenidos procedimentales, o metodológicos, sin renunciar a la formación o a la revisión de conceptos tales como el de sucesión; y la segunda, que una transcripción algebraica de las sucesiones, por medio de sus términos generales, enlaza bien con la idea de función y puede ayudar a comprender, e incluso a dar un limitado soporte algorítmico, a otros conceptos del cálculo.

No obstante, es en la visión global y constructiva de la sucesión en la que nos deberíamos basar a la hora de desarrollar algoritmos. No olvidemos que la descripción de las sucesiones y series mediante relaciones de recurrencia encuentra inmediata traducción en los esquemas sintácticos iterativos y recursivos que sirven de base a la mayoría de los lenguajes de programación.

Otro aspecto destacable de la formulación recurrente o global es que, a diferencia de la formulación con término general en que se procede por sustitución, aquí nos valemos de un factor constructivo medular en Matemáticas: el principio de inducción matemática. En opinión de Poincaré, el único soporte operativo del infinito.

Este principio al operar sobre las relaciones de recurrencia permite demostrar que cualquier término de la sucesión es calculable con ellas. La demostración es muy simple. Basta con representar mediante el predicado $C(n)$ el hecho de que el término $a_n$ es calculable. Con arreglo al principio de inducción si dicha condición se cumple (o dicho predicado se satisface) para $n = 1$ y además se cumple para $n = k + 1$ cuando se supone para $n = k$, entonces se cumple para cualquier número natural. En el caso considerado, $C(1)$ se cumple ya que $a_1$ viene dado y calculado en la primera relación. Si suponemos, como hipótesis de inducción, que $C(k)$ se cumple, esto equivale a que $a_k$ es calculable. Como tenemos una relación de recurrencia según la cual

$$a_{k+1} = \frac{a_k}{k}$$

y $a_k$ es calculable, entonces $a_{k+1}$ es calculable. Esto significa que $C(k + 1)$ se cumple y, por el principio de inducción, que la condición se cumple para todo $n$ natural.

Es curioso también observar el papel que la relación de recurrencia juega en el algoritmo, y cómo se obtiene el término genérico en lo que se denomina relación invariante de la iteración. Para el caso en que se pretende imprimir los 20 primeros términos de la sucesión que nos ocupa, presentaríamos el algoritmo

$a \leftarrow 1;\ n \leftarrow 1$
mientras $n \leq 20$ hacer
    escribir (a)
    $a \leftarrow a / n$
    $n \leftarrow n + 1$

Si $a$ e $n$ representan los valores que tras cada ejecución del ciclo alcanzan las variables, es fácil comprobar que siempre se verifica la relación

$$a = \frac{1}{(n-1)!}$$

que por esta razón llamamos invariante de la iteración, y que en este caso coincide formalmente con el término genérico de la sucesión. Este hecho pone de manifiesto que la convertibilidad entre término genérico y relaciones de recurrencia es posible, al igual que con la expresión funcional, cuando tan sólo contamos con el algoritmo.

Por otro lado, la estrecha relación existente entre el principio de inducción y las estructuras algorítmicas que dan soporte a los procesos repetitivos añade nuevo valor al enfoque algorítmico. Al respecto conviene tener en cuenta que si basamos la especificación algorítmica de una repetición en relaciones de recurrencia podremos verificar su corrección, es decir prodremos probar que se obtiene el cómputo deseado, con la inducción matemática.

Este hecho ha sido puesto de manifiesto por Dijkstra (1986) y otros autores, que entienden que una metodología de construcción de programas debe apoyarse en esquemas o patrones de pensamiento que contribuyan a la formación de estructuras expresivas de fácil verificación. Uno de los esquemas que consideran con preferencia es la inducción matemática, que a través de la correspondencia apuntada aseguraría la corrección de los programas.

Podemos ilustrar estas consideraciones con un nuevo ejemplo. Si consideramos el caso de la sucesión de sumas parciales de la serie natural, las relaciones recurrentes

$$\begin{cases} s_1 & = 1 \\ s_i & = s_{i-1} + i \end{cases}$$

ofrecen un método de construcción de valores, inmediatamente traducible a un algoritmo. En un caso finito, por ejemplo de 20 sumandos

$s \leftarrow 1;$
$i \leftarrow 2;$
mientras $i \leq 20$ hacer
    escribir (a)
    $s \leftarrow s + i$
    $i \leftarrow i + 1$

El algoritmo, tal y como hemos dicho, puede servir de base para ofrecer una prueba de la computabilidad de la serie y de que $s_n = \frac{n(n+1)}{2}$. Si $C(n)$ expresa ahora la validez de este hecho para el valor $n$, podemos razonar:

Se cumple $C(1)$ por la primera relación de recurrencia. Supongamos que se cumple $C(k)$, entonces

$$s_k = \frac{k(k+1)}{2}$$

y por la segunda relación de recurrencia

$$s_{k+1} = s_k + k + 1 = = \frac{k(k+1)}{2} + k + 1 = \frac{(k+1)(k+2)}{2}$$

es decir, $C(k + 1)$ es también válido. En virtud del principio de inducción podemos asegurar que $C(n)$ se cumple para cualquier número natural $n$.

En un artículo posterior el propio Dijkstra (1988) va más allá, y entiende que la expresión algorítmica puede constituir y ser adoptada como un medio normalizado de presentación de una amplia familia de demostraciones matemáticas. Esta propuesta

demostraría que no se ha extinguido todavía el espíritu de la polémica suscitada en los años 30 por la escuela intuicionista de Brouwer en favor de demostraciones, y consecuentemente de matemáticas, constructivas. Algunos autores (Gray 91) han propuesto, en este sentido, una revisión y reescritura en términos computacionales de ciertos teoremas clásicos del análisis matemático.

## Conclusiones

Entendemos en este trabajo que la adopción del empleo de algoritmos en la enseñanza secundaria es un obligado reconocimiento de la importancia que los métodos finitos están alcanzando en Matemáticas. Sin embargo, los algoritmos no deben quedar asociados a reglas aritméticas mecánicas, sino que es preciso adoptarlos como recurso lingüístico abierto y próximo al lenguaje natural en la resolución de problemas.

El progreso simbólico característico de la formación algebraica no debe verse impedido, ni obstaculizado por la expresión algorítmica.

Las diversas interpretaciones de la idea de variable en el ámbito algebraico e informático no parecen excluyentes, pero la elección de un formato algorítmico adecuado, en el seno de un lenguaje de programación, puede ampliar una visión restringida de las variables y apoyar la introducción de conceptos como el de función.

El recurso a los algoritmos como expresión de procesos repetitivos se puede relacionar con la idea de sucesión. De ella se ofrecen dos representaciones, una local y otra global. Observando las características del enfoque local a las sucesiones, tan sistemáticamente usado en las introducciones al Cálculo, no puede extrañar que la expresión funcional de magnitudes físicas y el estudio de sistemas dinámicos basados en tiempo continuo hayan sido históricamente el soporte principal del Cálculo infinitesimal. Y sin embargo raramente se enlaza la presentación de las sucesiones con este hecho.

Por otro lado, la formulación y cálculo basados en tiempo discreto, comunes en ciencias sociales o economía, hacen de la sucesión un instrumento matemático diferente, cuya significación es menos dependiente de la convergencia que del propio proceso de cálculo. En este marco el papel del algoritmo como expresión de la transformación efectiva de datos en resultados se revaloriza frente a la representación ideal de las situaciones problemáticas mediante relaciones funcionales.

## Bibliografía


Dijkstra E.W., Notes on Structured Programming, en Structured Programming Dahl, Dijkstra, Hoare (eds.), Academic P., 1972.

Dijkstra E.W., On a cultural gap, The Mathematical Intelligencer (1986) 48-52.

Goguen J.A., The logic of inexact concepts, Synthese (1968-69) 325-373.

Gray R., Computer Programs and Mathematical Proofs, The Mathematical Intelligencer (1991) 45-48.

Hoare C.A.R., An Axiomatic Basis for Computer Programming, Commun. of ACM (1969) 576-83.

Küchemann D.E., Algebra, en Children's understanding of Mathematics: 11-16 K. Hart (ed.), Murray, Londres, 1981.

Maurer S.B., Ralston A., Algorithms: Will cannot do discrete Mathematics without them, en Discrete Mathematics across the Curriculum K-12, NCTM, Reston U.S.A., 1991.

Mueller C., Teaching programming using values, World Comp. Congress, post. 82, Madrid, 1992.

Noss R., Constructing a Conceptual Framework for Elementary Algebra through LOGO Programming, Educational Studies in Mathematics (1986) 4.

Poincare H., Ciencia e hipótesis, Espasa-Calpe, Madrid, 1944.

Ralston A., Will Discrete Mathematics surpass Calculus in importance?, The College Mathematics Journal 15} (1984) 371-373.

Sutherland R.}. A longitudinal Study of the Development of Pupil's Algebraic Thinking in a LOGO Environment, Ph.D. Thesis, Univ. Londres.

Zellini P., Breve historia del infinito, Siruela, Madrid, 1991.