Dispense del corso di

LABORATORIO DI PROGRAMMAZIONE E CALCOLO

Marco Marfurt

Parte I: Analisi dell'errore propagato

0.1 Errori assoluti propagati

Siano x e y due numeri reali e siano $\overline{x}$ e $\overline{y}$ due loro *approssimazioni*; potremo quindi scrivere che

$$x = \overline{x} + \varepsilon_x \quad e \quad y = \overline{y} + \varepsilon_y \tag{1}$$

dove ε_x e ε_y sono due opportuni numeri reali; diremo che ε_x e ε_y sono gli *errori assoluti* di $\overline{x}$ e $\overline{y}$ rispetto ai *veri* valori di x e y.

Quando eseguiamo dei calcoli a partire dai *valori approssimati* $\overline{x}$ e $\overline{y}$ anzichè dai *valori esatti* x e y, gli *errori assoluti* ε_x e ε_y produrranno un errore sul risultato finale, che chiameremo *errore (assoluto) propagato*.

Quello che vogliamo ora studiare, è appunto come si propaga l'errore dei dati di partenza sul risultato finale nelle operazioni aritmetiche, cioè nella somma, nella sottrazione , nella moltiplicazione e nella divisione.

Cominciamo con l'esaminare il caso della somma: è ovvio che, sommando fra loro le (1), si avrà

$$x + y = (\overline{x} + \overline{y}) + (\varepsilon_x + \varepsilon_y)$$

e quindi l'errore propagato nella somma sarà dato semplicemente da

$$\varepsilon_{x+y} = \varepsilon_x + \varepsilon_y.$$

In modo del tutto analogo, sottraendo fra loro le (1), si avrà

$$x - y = (\overline{x} - \overline{y}) + (\varepsilon_x - \varepsilon_y)$$

e quindi l'errore propagato nella differenza sarà dato da

$$\varepsilon_{x-y} = \varepsilon_x - \varepsilon_y.$$

In forma più compatta potremo scrivere quindi

$$\varepsilon_{x\pm y} = \varepsilon_x \pm \varepsilon_y. \tag{2}$$

Passiamo ora all'operazione di prodotto; moltiplicando fra loro le (1), avremo in questo caso

$$xy = (\overline{xy}) + (\overline{x}\varepsilon_y + \overline{y}\varepsilon_x + \varepsilon_x\varepsilon_y)$$

e quindi l'errore propagato nel prodotto sarà dato da

$$\varepsilon_{xy} = \overline{x}\varepsilon_y + \overline{y}\varepsilon_x + \varepsilon_x\varepsilon_y, \tag{3}$$

e infine, dividendo fra loro le (1), otterremo questa volta

$$\frac{x}{y} = \frac{\overline{x} + \varepsilon_x}{\overline{y} + \varepsilon_y} = \frac{\overline{x}}{\overline{y}} + \frac{\overline{x} + \varepsilon_x}{\overline{y} + \varepsilon_y} - \frac{\overline{x}}{\overline{y}}$$

e quindi l'errore propagato nella divisione si potrà scrivere nella forma

$$\varepsilon_{\frac{x}{y}} = \frac{\overline{x} + \varepsilon_x}{\overline{y} + \varepsilon_y} - \frac{\overline{x}}{\overline{y}} = \frac{\overline{y}\varepsilon_x - \overline{x}\varepsilon_y}{\overline{y}y}. \tag{4}$$

Naturalmente noi non conosciamo i valori di ε_x e di ε_y, tuttavia in molti casi conosceremo almeno delle loro maggiorazioni; per esempio, supponiamo di conoscere un numero reale positivo ε tale che

$$|\varepsilon_x| \leq \varepsilon \quad e \quad |\varepsilon_y| \leq \varepsilon;$$

in questo caso dalla (2) segue che

$$|\varepsilon_{x \pm y}| \leq 2\varepsilon. \tag{5}$$

La (5) si può interpretare al modo seguente: *se conosciamo due numeri x e y con un errore assoluto al massimo ε , sommando o sottraendo fra loro le approssimazioni conosciute di x e y si ottiene un risultato che ha un errore assoluto al massimo 2ε .*

Si osservi che sia nel caso della somma che nel caso della differenza, la maggiorazione dell'errore che otteniamo è sempre 2ε , dal momento che, non sapendo a priori quali siano i segni di ε_x e di ε_y, gli errori possono, nel peggiore dei casi, sommarsi; la (5) è pertanto una *maggiorazione pessimistica.*

Passiamo ora a vedere che cosa succede nel prodotto; dalla (3) si ottiene la disuguaglianza:

$$|\varepsilon_{xy}| \leq (|\overline{x}| + |\overline{y}|)\varepsilon + \varepsilon^2 \tag{6}$$

Analogamente a quanto fatto per la disuguaglianza (5) possiamo interpretare la disuguaglianza (6) al modo seguente: *se conosciamo due numeri x e y con un errore assoluto al massimo ε , moltiplicando fra loro le approssimazioni conosciute di x e y si ottiene un risultato che ha un errore assoluto al massimo*

$$(|\overline{x}| + |\overline{y}|)\varepsilon + \varepsilon^2.$$

Diversamente da quanto succedeva nel caso di somma e differenza, nel caso del prodotto l'errore assoluto sul risultato finale *potrebbe* essere molto più grande di ε (e quindi molto più grande degli errori assoluti sui dati di partenza) nel caso in cui o $|\overline{x}|$ o $|\overline{y}|$ o tutti e due fossero molto grandi.

Per finire, vediamo cosa succede nel caso della divisione; dalla (4) si ottiene la disuguaglianza:

$$|\varepsilon_{\frac{x}{y}}| \leq \frac{(|\overline{x}| + |\overline{y}|)\varepsilon}{|\overline{y}y|} \tag{7}$$

che, interpretata come abbiamo fatto in precedenza ci dice che *se conosciamo due numeri x e y con un errore assoluto al massimo ε , dividendo fra loro le approssimazioni conosciute di x e y si ottiene un risultato che ha un errore assoluto al massimo*

$$\frac{(|\overline{x}| + |\overline{y}|)\varepsilon}{|\overline{y}y|}.$$

Anche qui, come nel caso del prodotto, l'errore assoluto sul risultato finale *potrebbe* essere molto più grande di ε (e quindi molto più grande degli errori assoluti sui dati di partenza) nel caso in cui $|\overline{x}|$ fosse molto grande oppure nel caso in cui $|\overline{y}|$ (o y) fosse molto piccolo.

In base a ciò che abbiamo visto finora, *sembrerebbe* che le operazioni di somma e di sottrazione siano *operazioni buone*, mentre quelle di moltiplicazione e di divisione siano *operazioni cattive*, dal momento che l'errore assoluto sul risultato del calcolo rimane dello stesso ordine di grandezza degli errori assoluti sui dati di partenza nelle prime due operazioni, mentre può diventare molto più grande nelle seconde due.

Tuttavia, come vedremo nel prossimo paragrafo, le cose non stanno esattamente in questo modo.

0.2 Errori relativi propagati

Supponiamo di conoscere un numero $\overline{x}$ che approssima un certo numero x con un errore assoluto ε_x e un numero $\overline{y}$ che approssima un certo numero y con un errore assoluto ε_y dei quali sappiamo che

$$|\varepsilon_x| \leq 0.000001 \quad e \quad |\varepsilon_y| \leq 1000000; \tag{8}$$

e chiediamoci: quale fra i due numeri x e y *conosciamo meglio*?

Inizialmente saremmo portati a dire che conosciamo meglio x che y, tuttavia, riflettendo meglio è facile rendersi conto che la risposta corretta dipende molto dall'*ordine di grandezza* dei due numeri $\overline{x}$ e $\overline{y}$; infatti, se per esempio fosse

$$\overline{x} = 0.000000000234567 \quad e \quad \overline{y} = 2345678912, \tag{9}$$

tenendo conto delle (8), potremmo dire che le tre prime cifre significative di $\overline{y}$ (cioè 234) devono coincidere con le tre prime cifre significative di y, mentre viceversa, per quanto riguarda x, non solo non possiamo dire la stessa cosa, ma addirittura la prima cifra significativa di x *potrebbe essere in una posizione diversa* dalla prima cifra significativa di $\overline{x}$ e quindi $\overline{x}$ *potrebbe essere addirittura di un ordine di grandezza diverso* da quello di x. É chiaro quindi che in questo caso conosceremmo meglio il numero y che il numero x.

Queste osservazioni ci portano ad introdurre il concetto di *errore relativo*, che definiremo al modo seguente: sia x un numero reale e sia $\overline{x}$ una sua approssimazione nota con un errore assoluto ε_x, per cui sarà

$$x = \overline{x} + \varepsilon_x$$

chiameremo *errore relativo* di $\overline{x}$ il numero

$$\varrho_x = \frac{\varepsilon_x}{\overline{x}}.$$

Se usiamo l'errore relativo per i numeri $\overline{x}$ e $\overline{y}$ dell'esempio (8) e (9), vediamo subito che, pur essendo

$$|\varepsilon_x| \ll |\varepsilon_y|$$

risulta invece che

$$|\varrho_y| \ll |\varrho_x|$$

il che mostra appunto come sia molto più significativo l'errore relativo che l'errore assoluto.

Vediamo ora come si comportano gli errori relativi nelle operazioni aritmetiche, cioè vediamo di studiare la questione dell'errore di propagazione per quanto riguarda l'errore relativo. Prima di tutto supponiamo che x e y siano due numeri reali di cui conosciamo le approssimazioni $\overline{x}$ e $\overline{y}$ con errori assoluti ε_x e ε_y rispettivamente, per cui sarà

$$x = \overline{x} + \varepsilon_x \quad e \quad y = \overline{y} + \varepsilon_y$$

e quindi gli errori relativi su $\overline{x}$ e $\overline{y}$ saranno rispettivamente:

$$\varrho_x = \frac{\varepsilon_x}{\overline{x}} \quad e \quad \varrho_y = \frac{\varepsilon_y}{\overline{y}}; \tag{10}$$

dalla (2) e dalla definizione di errore relativo segue allora che

$$\varrho_{x\pm y} = \frac{\varepsilon_{x\pm y}}{\overline{x} \pm \overline{y}} = \frac{\varepsilon_x \pm \varepsilon_y}{\overline{x} \pm \overline{y}} \tag{11}$$

e se anche qui supponiamo che sia

$$|\varepsilon_x| \leq \varepsilon \quad e \quad |\varepsilon_y| \leq \varepsilon,$$

otteniamo infine che

$$|\varrho_{x\pm y}| \leq \frac{2\epsilon}{|\overline{x} \pm \overline{y}|}. \tag{12}$$

Al contrario di quello che succedeva nella (5) per l'errore assoluto, qui vediamo che l'errore relativo sul risultato della somma o della sottrazione di x e y può diventare molto più grande degli errori relativi sui dati di partenza se la quantità $|\overline{x} \pm \overline{y}|$ è più piccola di ε; in sostanza quindi, l'operazione di somma o di differenza di due numeri $\overline{x}$ e $\overline{y}$ può dar luogo ad una amplificazione (talvolta disastrosa) dell'errore relativo sui dati nel caso in cui $\overline{x}$ e $\overline{y}$ siano due numeri *molto vicini fra loro in valore assoluto* e contemporaneamente l'operazione $\pm$ sia una *sottrazione effettiva*.

Passiamo ora a vedere cosa accade dell'errore relativo nel prodotto e nella divisione; dividendo la (3) per $\overline{xy}$ otteniamo

$$\varrho_{xy} = \varrho_x + \varrho_y + \varrho_x\varrho_y \tag{13}$$

e analogamente, dividendo la (4) per $\overline{x}/\overline{y}$ otteniamo

$$\varrho_{x/y} = \frac{\overline{y}}{y}(\varrho_x - \varrho_y) = \frac{1}{1+\varrho_y}(\varrho_x - \varrho_y). \tag{14}$$

Supponiamo ora di conoscere due buone approssimazioni $\overline{x}$ e $\overline{y}$ di x e y, cioè quindi supponiamo che sia

$$|\varrho_x| \ll 1 \quad e \quad |\varrho_y| \ll 1;$$

è facile allora vedere che dovrà anche essere

$$|\varrho_x\varrho_y| \ll |\varrho_x| \quad e \quad |\varrho_x\varrho_y| \ll |\varrho_y|$$

e quindi dalla (13) segue che, nel caso più sfortunato in cui gli errori sui dati si sommano, avremo comunque che

$$|\varrho_{xy}| \simeq |\varrho_x| + |\varrho_y|, \tag{15}$$

mentre, per quanto riguarda la (14), nelle stesse ipotesi sarà

$$\frac{1}{1+\varrho_y} \simeq 1$$

e quindi varrà di nuovo la (15) anche per la divisione.

La (15) ci dice in sostanza che le operazioni di prodotto e divisione si *comportano bene* per quanto riguarda la propagazione dell'errore relativo, infatti in entrambe le operazioni l'errore relativo sul risultato è al massimo la somma degli errori relativi sui dati.

Vediamo ora alcuni semplici esempi di algoritmi di calcolo che illustrano quanto detto finora sugli errori propagati nelle operazioni aritmetiche.

0.3 Risolviamo una equazione di secondo grado

Supponiamo di dover calcolare le radici dell'equazione

$$ax^2 + bx + c = 0 \tag{16}$$

e supponiamo inoltre che sia

$$\Delta = b^2 - 4ac > 0$$

per cui la (16) ammette esattamente due radici reali e distinte; come tutti sappiamo, le due radici di (16) possono essere facilmente calcolate dalla formula

$$\frac{-b \pm \sqrt{\Delta}}{2a}. \tag{17}$$

Se b è un numero reale tale che

$$|b| \gg |a| \quad e \quad |b| \gg |c|$$

si avrà che

$$\Delta = b^2 - 4ac \simeq b^2$$

e quindi

$$|b| \simeq \sqrt{\Delta}.$$

Dalla formula (17) segue allora che, per calcolare le due radici reali di (16), dobbiamo sommare e sottrarre due numeri molto vicini fra loro; poichè quando calcoliamo $\sqrt{\Delta}$ introdurremo necessariamente un errore dovuto all'operazione di *radice quadrata*, da quanto visto nei precedenti paragrafi, si deduce che questo errore potrà venire amplificato (anche in maniera *disastrosa*) quando eseguiamo la sottrazione.

Supponiamo, per esempio, di voler calcolare le due radici reali dell'equazione di secondo grado

$$x^2 - 2^k x + 1 = 0 \tag{18}$$

con k intero positivo. L'equazione (18) ha due radici reali x_1 e x_2 che, per valori di k grandi, sono molto prossime rispettivamente a 2^k e a $\frac{1}{2^k}$; tuttavia, se utilizziamo un programma C++ per calcolare le radici di (18) usando la formula (17), per $k = 13$ otteniamo

$$x_1 = 8.192 \cdot 10^3 \quad e \quad x_2 = 1.220703125 \cdot 10^{-4}$$

mentre per $k = 14$ otteniamo

$$x_1 = 1.6384 \cdot 10^4 \quad e \quad x_2 = 0.000 \cdot 10^0.$$

Se teniamo conto del fatto che le radici esatte della nostra equazione , nel primo caso (cioè per $k = 13$) sono

$$x_1 = 8.1919998779296857 \cdot 10^3 \quad e \quad x_2 = 1.2207031431898946 \cdot 10^{-4}$$

mentre nel secondo caso (cioè per $k = 14$) sono

$$x_1 = 1.6383999938964844 \cdot 10^4 \quad e \quad x_2 = 6.1035156477373675 \cdot 10^{-5}$$

vediamo che nel primo caso la formula (17) ci ha fornito delle buone approssimazioni di entrambe le radici, mentre nel secondo caso solo la prima radice viene approssimata correttamente; la seconda radice diventa zero e su di essa perdiamo ogni informazione, cioè non ne conosciamo più neanche l'ordine di grandezza. Ciò è dovuto essenzialmente al fenomeno della amplificazione dell'errore relativo nella sottrazione fra due numeri molto vicini (infatti è proprio la radice x_2 quella che si ottiene nella (17) con una sottrazione).

Quando un algoritmo di calcolo presenta questo tipo di comportamento, cioè è tale che un piccolo errore relativo sui dati o nei calcoli può portare ad un grande errore relativo sul risultato finale, si dice che esso è *instabile* (e viceversa diremo *stabile* un algoritmo che non amplifica nel risultato finale gli errori relativi sui dati o nei calcoli).

Abbiamo quindi scoperto che l'algoritmo per calcolare le radici di una equazione di secondo grado fornito dalle formule (17) può, in alcuni casi, essere *instabile*; ci si pone il problema di vedere se esista un qualche altro algoritmo che risulti sempre *stabile*: a questo scopo, osserviamo che, se nelle

formule (17) moltiplichiamo numeratore e denominatore per $-b \mp \sqrt{\Delta}$, con qualche semplificazione otteniamo le formule

$$\frac{2c}{-b \mp \sqrt{\Delta}}. \tag{19}$$

É poi anche facile constatare che la radice che si ottiene nelle formule (19) utilizzando il segno + è la stessa radice che si ottiene nelle formule (17) utilizzando il segno −, e analogamente la radice che si ottiene nelle formule (19) utilizzando il segno − è la stessa radice che si ottiene nelle formule (17) utilizzando il segno +; ciò significa quindi che, se usiamo le formule (19) per calcolare le radici dell'equazione (18), otterremo un risultato opposto a quello precedente, cioè otterremo sempre buone approssimazioni della radice più piccola x_2, mentre (per k grande) otterremo cattive approssimazioni della radice più grande x_1; si tratta quindi ancora di un algoritmo che può essere *instabile*; tuttavia ora abbiamo una soluzione del nostro problema, che consiste nell'usare contemporaneamente le formule (17) e le formule (19) selezionando il segno + o il segno − in modo che non si debba mai fare una sottrazione, in questo modo otterremo entrambe le radici senza pericolo di amplificazione dell'errore relativo.

Ecco quindi come sarà un algoritmo *stabile* per calcolare le radici dell'equazione di secondo grado (16):

calcola $s = \sqrt{b^2 - 4ac}$

se $b < 0$ *calcola le radici con le formule*

$$x_1 = \frac{-b + s}{2a} \quad , \quad x_2 = \frac{2c}{-b + s}$$

se $b \geq 0$ *calcola le radici con le formule*

$$x_1 = \frac{-b - s}{2a} \quad , \quad x_2 = \frac{2c}{-b - s}.$$

0.4 Quanto vale π ?

In questo paragrafo ci proponiamo di determinare con grande precisione il valore del numero trascendente π. Un modo concettualmente molto semplice per fare questo consiste nel costruire due poligoni regolari con n lati rispettivamente inscritto e circoscritto ad una circonferenza di raggio unitario; i semiperimetri dei due poligoni saranno due approssimazioni rispettivamente per difetto e per eccesso della lunghezza della semicirconferenza che è ovviamente π. Inoltre, al crescere del numero n dei lati dei due poligoni, queste

approssimazioni diventeranno sempre più stringenti, permettendoci così, in teoria, di determinare il valore di π con una precisione sempre più grande.

Ho detto *in teoria*, perchè ovviamente avremo delle limitazioni: per esempio, una ovvia limitazione sarà dovuta al fatto che noi rappresentiamo un numero reale nel calcolatore con una precisione finita oltre la quale sembra difficile poter andare; un'altra limitazione potrebbe essere legata al fatto che l'algoritmo che usiamo potrebbe non risultare *stabile* e quindi potrebbe vanificare i nostri sforzi.

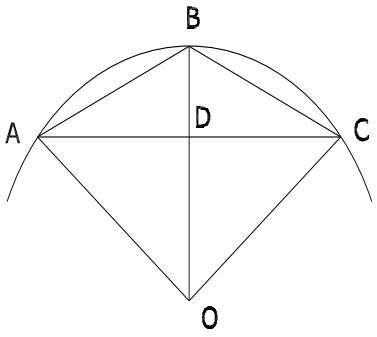

Figura 1:

Per poter realizzare il nostro algoritmo, dobbiamo prima di tutto avere una formula per calcolare la lunghezza del lato di un poligono regolare (inscritto o circoscritto) con n lati. A questo scopo, osserviamo il disegno in figura (1): se supponiamo che il segmento $\overline{AC}$ sia il lato del poligono regolare con n lati inscritto nel cerchio di raggio unitario, il segmento $\overline{AB}$ sarà allora il lato del poligono regolare con $2n$ lati inscritto nel cerchio di raggio unitario. Se poniamo per brevità

$$l = \overline{AC} \quad e \quad l' = \overline{AB}$$

non è difficile dimostrare che

$$l' = \sqrt{2 - \sqrt{4 - l^2}} \tag{20}$$

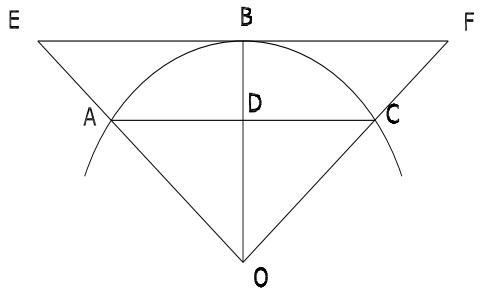

Figura 2:

Se osserviamo ora il disegno in figura (2) e se supponiamo sempre che il segmento $\overline{AC}$ sia il lato del poligono regolare con n lati inscritto nel cerchio di raggio unitario, il segmento $\overline{EF}$ sarà allora il lato del poligono regolare con n lati circoscritto al cerchio di raggio unitario. Se poniamo per brevità

$$l = \overline{AC} \quad e \quad L = \overline{EF}$$

anche qui non è difficile dimostrare che

$$L = \frac{2l}{\sqrt{4 - l^2}} \tag{21}$$

Le formule (20) e (21) sono tutto ciò che ci serve per costruire un algoritmo per il calcolo di π; infatti, se indichiamo con l_n il *lato del poligono regolare con* 2^n *lati inscritto nel cerchio di raggio unitario*, dalla (20) segue subito che il *lato* l_{n+1} *del poligono regolare con* 2^{n+1} *lati inscritto nel cerchio di raggio unitario* sarà fornito dalla formula

$$l_{n+1} = \sqrt{2 - \sqrt{4 - l_n^2}}. \tag{22}$$

Poichè il lato del quadrato inscritto nel cerchio di raggio unitario è dato da

$$l_2 = \sqrt{2}, \tag{23}$$

le (22) e (23) permettono facilmente di costruire la successione $\{l_n\}$. A questo punto basterà moltiplicare ciascun l_n per 2^{n-1} per ottenere il semiperimetro del poligono regolare con 2^n lati inscritto nel cerchio di raggio unitario.

Infine, se indichiamo con L_n il *lato del poligono regolare con* 2^n *lati circoscritto al cerchio di raggio unitario*, dalla (21) segue che

$$L_n = \frac{2l_n}{\sqrt{4 - l_n^2}} \tag{24}$$

e moltiplicando ciascun L_n per 2^{n-1} si ottiene il semiperimetro del poligono regolare con 2^n lati circoscritto al cerchio di raggio unitario.

Le (22), (23) e (24) consentono di costruire facilmente in modo ricorsivo le due successioni $\{l_n\}$ e $\{L_n\}$ che, al crescere di n ci daranno approssimazioni per difetto e per eccesso di π sempre più stringenti.

Se realizziamo l'algoritmo ora descritto tramite un programma C++ e ci facciamo stampare le approssimazioni per difetto e per eccesso di π, possiamo facilmente osservare che accade un fenomeno curioso: per i primi valori di n le due successioni $\{l_n\}$ e $\{L_n\}$ si avvicinano effettivamente al valore di π, ma da un certo punto in poi, al crescere di n i valori delle due successioni cominciano ad assumere valori strani e sempre più distanti da π. Siamo evidentemente di nuovo di fronte ad un algoritmo *instabile*; vediamo di capirne il perchè.

Se osserviamo la formula (22) (che è la formula basilare del nostro algoritmo), vediamo subito che all'interno della radice quadrata più esterna noi dobbiamo effettuare la sottrazione di 2 e di $\sqrt{4 - l_n^2}$; poichè ovviamente deve essere

$$\lim_{n\to\infty} l_n = 0$$

si avrà di conseguenza che, per n grande risulterà

$$\sqrt{4 - l_n^2} \simeq 2$$

e quindi di nuovo l'instabilità nasce dalla sottrazione di due numeri molto vicini fra loro. É possibile ovviare a questo inconveniente e modificare il nostro algoritmo in modo da renderlo stabile? Per fortuna anche in questo caso la soluzione del problema è semplice e assomiglia molto a quella che abbiamo escogitato per rendere stabile l'algoritmo che calcola le radici di una equazione di secondo grado.

Partiamo infatti dalla (22) e moltiplichiamola e dividiamola per la quantità

$$\sqrt{2+\sqrt{4-l_n^2}};$$

con qualche semplificazione otteniamo allora la formula alternativa

$$l_{n+1} = \frac{l_n}{\sqrt{2+\sqrt{4-l_n^2}}}. \tag{25}$$

La formula (25), insieme con le (23) e (24), ci permette di nuovo di calcolare le successioni $\{l_n\}$ e $\{L_n\}$ e di approssimare π, ma questa volta senza mai dover sottrarre numeri molto vicini fra loro. Se realizziamo l'algoritmo basato sulla (25) tramite un programma C++ e ci facciamo stampare le approssimazioni per difetto e per eccesso di π, possiamo facilmente osservare che questa volta tutto funzionerà alla perfezione e le due successioni continueranno ad approssimare π sempre meglio al crescere di n, cioè l'algoritmo così modificato è un algoritmo stabile.