TEILWEISE ASYNCHRONE ALGORITHMEN

FRANK LANGBEIN

Literatur: D. Bersekas, J. Tsitsiklis: Parallel and distributed computatoin, pp. 481 – 489
URI: 'http://www.langbein.org/research/parallel/'

- Modell teilweiser asynchroner Algorithmen
- Konvergenzverhalten
- Satz von Lyapunov

Modell teilweiser asynchroner Algorithmen

Entspricht im wesentlichen dem vollständig asynchroner Algorithmen:

$X_1, X_2, \ldots, X_n$	Teilmengen euklidischer Räume
$X = X_1 \times X_2 \times \cdots \times X_n$	
$x = (x_1, x_2, \ldots, x_n) \in X$	mit $x_i \in X_i$ für $i = 1, \ldots, n$
$f_i : X \to X_i$	$i = 1, \ldots, n$

Wir betrachten folgende asynchrone Iteration

$$x_i := f_i(x), \qquad i = 1, \ldots, n$$

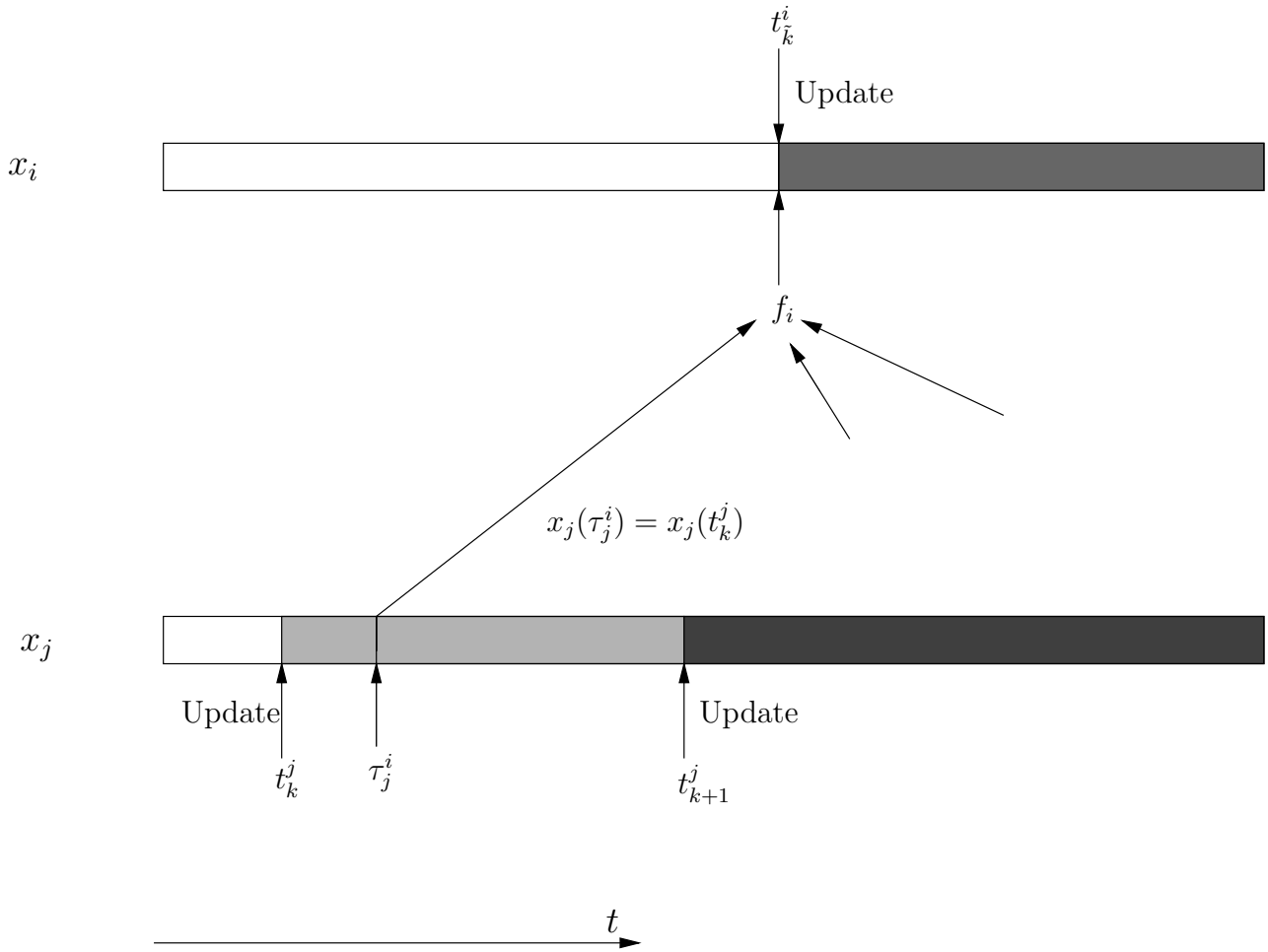
Diese Iteration ist vollständig bestimmt durch

(a) Anfangsbedingungen $x(t) \in X$ für *jedes* $t \leq 0$
(b) $T^i = \{t_0^i, t_1^i, t_2^i, \ldots\}$ mit $0 \leq t_0^i < t_1^i < t_2^i < \ldots$, den Zeitpunkten zu denen ein Update von x_i erfolgt $(i = 1, \ldots, n)$.
(c) $\tau_j^i(t)$ für jedes $i, j \in \{1, \ldots, n\}$ und $t \in T^i$, welches angibt wie veraltet der Wert von x_j ist, der für das Update von x_i zur Zeit t verwendet wurde; $0 \leq \tau_j^i(t) \leq t$.
Um die Analyse der Algorithmen zu vereinfachen sind für $\tau_j^i(t)$ auch negative Werte erlaubt, deshalb werden Anfangsbedingungen auch für negative t gefordert.

Der asyncrhone Algorithmus für $t \geq 0$ wird beschrieben durch

$$\begin{aligned} x_i(t+1) &= x_i(t) && \text{für } t \notin T^i \\ x_i(t+1) &= f_i(x_1(\tau_1^i(t)), x_2(\tau_2^i(t)), \ldots, x_n(\tau_n^i(t))) && \text{für } t \in T^i \end{aligned}$$

Date: Feb-09-1996.



Scenario: Spezielle Wahl von T^i und $\tau_j^i(t)$
Für jedes festes Scenario ist $x(t)$ für $t > 0$ eindeutig durch die Anfangsbedingungen bestimmt. Jedoch können die Werte von $x(t)$ für verschiedene Scenarien bei gleichen Anfangsbedingungen stark voneinander abweichen.

Teilweise Asynchronität. Es existiert eine positive ganze Zahl B, so daß

(a) für jedes i und $t \geq 0$ mindestens ein Element der Menge $\{t, t+1, \ldots, t+B-1\}$ zu T^i gehört.
(b) $t - B < \tau_j^i(t)$ für alle i und j und $t \in T^i$
(c) $\tau_i^i(t) = t$ für alle i und $t \in T^i$

Zu (a): Jeder Prozessor führt wenigstens ein Update während eines beliebigen Zeitintervalls der Länge B durch.

Zu (b): Jeder Prozessor verwendet Informationen die nicht älter als B Zeiteinheiten sind.

Zu (c): Beim Update von x_i zum Zeitpunkt $t+1$ wird $x_i(t)$ verwendet. In 'Message Passing Systems' ist dies automatisch erfüllt, da der i-te Prozessor den Wert der Variablen x_i speichert und der einzigste Prozessor ist, der x_i verändern kann. In 'Shared Memory Systems' ist dies nicht unbedingt erfüllt, kann aber leicht erreicht werden.
Es kann gezeigt werden, daß ein sonst konvergierender Algorithmus nicht mehr konvergiert, wenn (a) und (b) gelten, aber (c) verletzt ist.

Beispiel 1

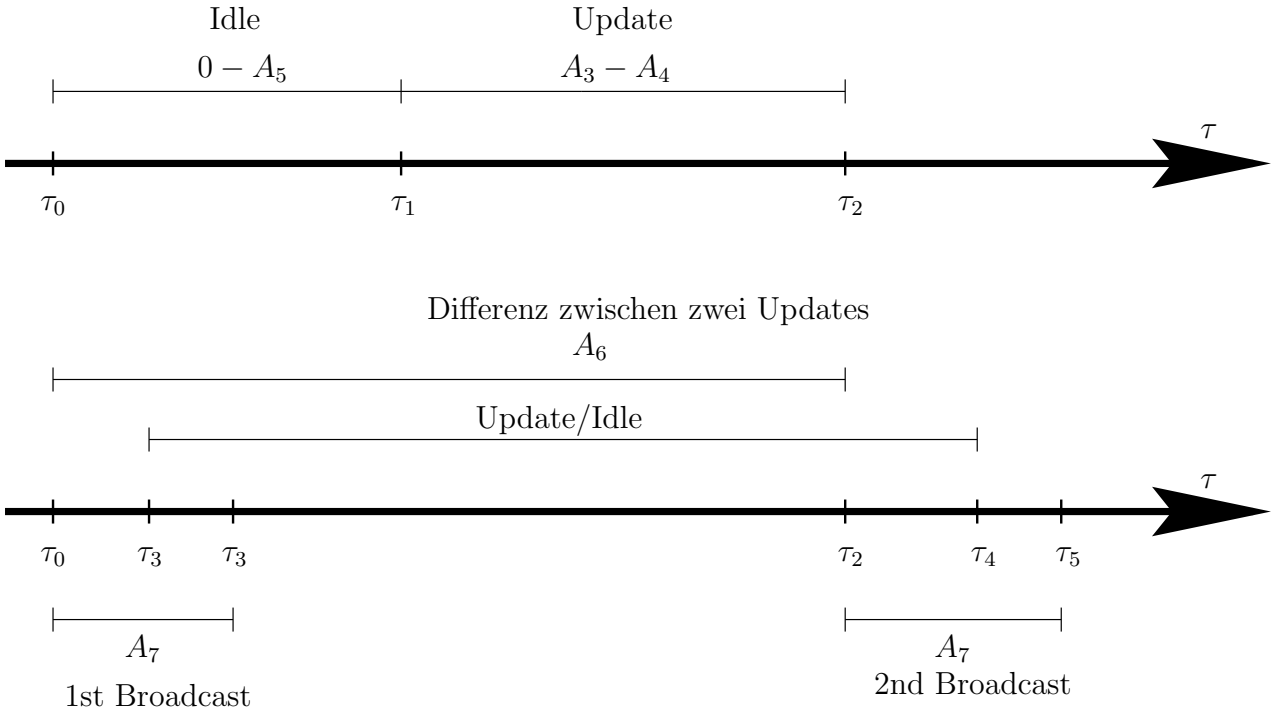
Die Zeitvariable t muß den Prozessoren nicht zugänglich sein. Sie muß auch nicht der realen Zeit entsprechen. t kann z.B. ein Index für bestimmt Ereignisse wie Updates sein.
Wir betrachten dazu ein 'Message Passing System' mit n Prozessoren. Jeder Prozessor hat seine eigene lokale Uhr.

Reale (globale) Zeit, den Prozessoren unbekannt:	τ
Lokale Zeit des Prozessors i zur realen Zeit τ:	$t_i(\tau)$
Positive reelle Konstanten (siehe unten):	A_1, A_2
Lokale Zeiteinheiten die ein Update benötigt:	A_3 bis A_4
Maximale idle Zeit in lokalen Einheiten:	A_5
Maximale lokale Zeitdifferenz zwischen zwei Updates von x_i:	A_6
Maximale globale Zeit für eine Übertragung:	A_7

Die einzelnen Prozessoruhren können nicht beliebig schnell oder langsam sein:

$$A_1|\tau - \tau'| \leq |t_i(\tau) - t_i(\tau')| \leq A_2|\tau - \tau'| \quad \forall \tau, \tau', i$$

t sei eine Indexvariable, die um 1 erhöht wird wenn irgendein Prozessor ein Update ausführt, $t(\tau)$ ist der Wert der Indexvariable t zum Zeitpunkt τ. t wird hier als Zeitvariable für das Modell verwendet. Unter diese Vorrausetzungen sind (a) und (b) der teilweisen Asynchronität erfüllt.



(a):

$$A_3 \leq |t_i(\tau_2) - t_i(\tau_0)| \leq A_4 + A_5 \qquad A_1|\tau_2 - \tau_0| \leq |t_i(\tau_2) - t_i(\tau_0)| \leq A_2|\tau_2 - \tau_0|$$

$$\Rightarrow \frac{A_3}{A_2} \leq |\tau_2 - \tau_0| \leq \frac{A_4 + A_5}{A_1}, \qquad U = \left\lceil \frac{A_4 + A_5}{A_1} \frac{A_3}{A_2} \right\rceil (n-1) \text{ Updates maximal in } \tau_0 - \tau_2$$

$$\Rightarrow t(\tau_0) + U \geq t(\tau_2) \quad \Rightarrow (a)$$

(b):

$$\tau_5 - \tau_2 \leq A_7, \qquad t_i(\tau_2) - t_i(\tau_0) \leq A_6 \qquad \Rightarrow \tau_5 - \tau_0 \leq \frac{A_6}{A_1} + A_7$$

$$\Rightarrow t(\tau_5) - t(\tau_0) \leq \left\lceil \left(\frac{A_6}{A_1} + A_7 \right) \frac{A_2}{A_3} \right\rceil n \qquad \Rightarrow (b)$$

Beispiel 2: Konvergenz für beliebiges B

Wir betrachten die lineare Iteration

$$x := f(x) = Ax, \qquad A = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Die Fixpunkte von $f(x)$ sind $X^* = \{(x_1, x_2) \in \mathbb{R} : x_1 = x_2\}$. Die synchrone Iteration $x(t+1) = Ax(t)$ konvergiert gegen $x = (y, y), y = \frac{x_1(0)+x_2(0)}{2}$ in einem Schritt.

Asynchrone Iteration

Wir verwenden zwei Prozessoren $i = 1, 2$. Zu jedem Zeitschritt erfolgt ein Update der Variablen x_i, also $T^i = \{1, 2, 3, \dots\}$. Zu bestimmten Zeiten $t_1, t_2, t_3, \dots$ überträgt jeder Prozessor den aktuellen Wert von x_i. Dabei nehmen wir an, daß keine Kommunikationsverzögerung auftritt. Somit erhalten wir

$$\begin{aligned} x_1(t+1) &= \tfrac{x_1(t)}{2} + \tfrac{x_2(t_k)}{2} \\ x_2(t+1) &= \tfrac{x_1(t_k)}{2} + \tfrac{x_2(t)}{2} \end{aligned} \qquad t_k \le t < t_{k+1}$$

Also

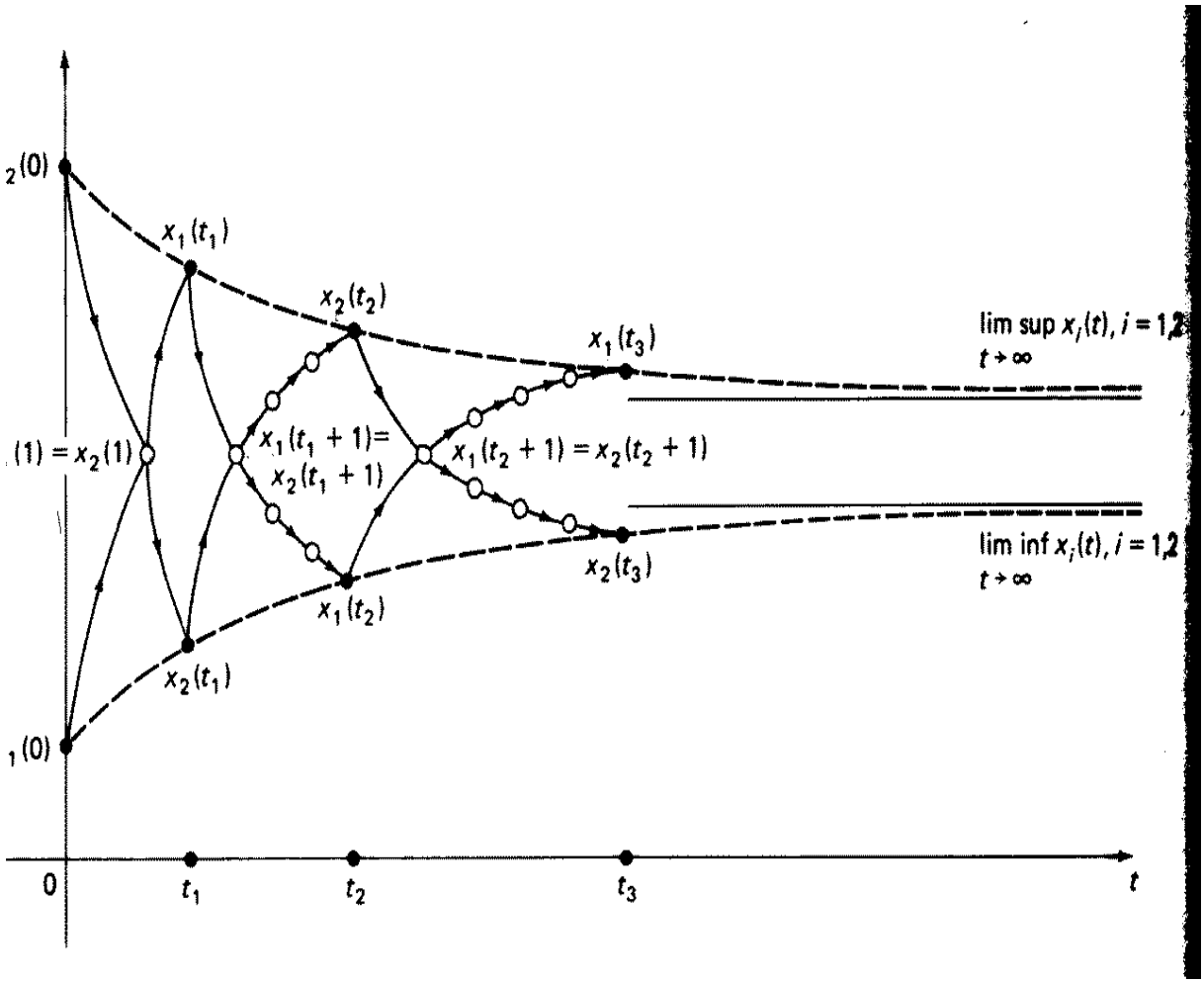
$$\begin{aligned} x_1(t_{k+1}) &= \frac{x_1(t_{k+1}-1)}{2} + \frac{x_2(t_k)}{2} \\ &= \frac{x_1(t_{k+1}-2)}{2^2} + \left(\frac{1}{2} + \frac{1}{2^2}\right) x_2(t_k) \\ &= \dots \\ &= \left(\frac{1}{2}\right)^{t_{k+1}-t_k} x_1(t_k) + \left(\frac{1}{2} + \frac{1}{2^2} + \dots + \frac{1}{2^{t_{k+1}-t_k}}\right) x_2(t_k) \\ &= \left(\frac{1}{2}\right)^{t_{k+1}-t_k} x_1(t_k) + \left(1 - \left(\frac{1}{2}\right)^{t_{k+1}-t_k}\right) x_2(t_k) \\ x_2(t_{k+1}) &= \left(\frac{1}{2}\right)^{t_{k+1}-t_k} x_2(t_k) + \left(1 - \left(\frac{1}{2}\right)^{t_{k+1}-t_k}\right) x_1(t_k) \end{aligned}$$

$$\begin{aligned} |x_2(t_{k+1}) - x_1(t_{k+1})| &= \left(1 - 2\left(\frac{1}{2}\right)^{t_{k+1}-t_k}\right) |x_2(t_k) - x_1(t_k)| \\ &= (1-\epsilon_k)|x_2(t_k) - x_1(t_k)| \qquad \text{mit } \epsilon_k = 2\left(\frac{1}{2}\right)^{t_{k+1}-t_k} \end{aligned}$$

Damit nimmt $|x_2(t_k) - x_1(t_k)|$ ab. Hieraus folgt aber noch nicht die Konvergenz gegen ein Element aus X^*, denn hierzu muß $\prod_{k=1}^{\infty}(1-\epsilon_k) = 0$ erfüllt sein.
Es gilt z.B.

$$\prod_{k=2}^{\infty}(1-k^{-2}) = \prod_{k=2}^{\infty}\left(\frac{k-1}{k}\,\frac{k+1}{k}\right) = \left(\frac{1}{2}\,\frac{3}{2}\right)\left(\frac{2}{3}\,\frac{4}{3}\right)\left(\frac{3}{4}\,\frac{5}{4}\right)\cdots = \frac{1}{2} > 0$$

Wählt man also $t_{k+1} - t_k$ groß genug, so daß $\epsilon_k < k^{-2}$, dann konvergiert x nicht (vgl. Fig.). Hieraus folgt, daß die Iteration im vollständig asynchronen Fall nicht konvergiert.



t	0	1	$t_1 = 2$	3	4	5	$t_2 = 6$	7	$\cdots$
$x_1(t)$	4	10	13	10	8.5	7.75	7.375	10	
$x_2(t)$	16	10	7	10	11.5	12.25	12.625	10	

Für die teilweise asynchrone Iteration muß $t_{k+1} - t_k \leq B$ sein. Also $\epsilon_k > \left(\frac{1}{2}\right)^B$ und damit

$$0 \leq \prod_{k=1}^{\infty}(1-\epsilon_k) < \prod_{k=1}^{\infty}\left(1-\frac{1}{2^B}\right) = \lim_{n\to\infty}\left(1-\frac{1}{2^B}\right)^n = 0$$

Damit konvergiert $|x_1(t_k) - x_2(t_k)|$ gegen 0 und somit auch $x(t_k)$ gegen ein Element aus X^*.

- Obige Iteration is ein Beispiel für einen Algorithmus der nicht vollständig asynchron konvergiert, aber für jedes B teilweise asynchron konvergiert.
- Für solche Algorithmen gibt es normalerweise eine Funktion, die den Abstand von X^* mißt und mit der Zeit kleiner wird.
- Der Faktor mit dem diese Abstandsfunktion abnimmt hängt jedoch von den Kommunikationsverzögerungen ab. Wir müssen daher annehmen, daß diese begrenzt sind um zu garantieren, daß die Abstandsfunktion mit einer festen Rate fällt.

Beispiel 3: Konvergenz für kleine B

Wir betrachten nun die Iteration für den Gradienten Algorithmus zur Minimierung von $\frac{1}{2}x^TAx$ mit der Schrittweite γ.

$$x := x - \gamma Ax, \qquad A = \begin{pmatrix} 1+\epsilon & 1 & 1 \\ 1 & 1+\epsilon & 1 \\ 1 & 1 & 1+\epsilon \end{pmatrix} = \epsilon I + ee^T, \qquad 0 < \epsilon < 1$$

Die synchrone Iteration konvergiert für kleine γ. Die vollständig asynchrone Iteration konvergiert nicht.

Teilweise Asynchrone Iteration

Pro Zeiteinheit wird von jedem Prozessor ein Update ausgeführt. Die Variablen werden alle B Zeiteinheiten übertragen, also $t_k = kB$. Dies ergibt

$$\begin{array}{rcll} x_1(t+1) & = & (1-\gamma(1+\epsilon))x_1(t) - \gamma(x_2(t_k) + x_3(t_k)) & \\ x_2(t+1) & = & (1-\gamma(1+\epsilon))x_2(t) - \gamma(x_1(t_k) + x_3(t_k)) & t_k \le t < t_{k+1} \\ x_3(t+1) & = & (1-\gamma(1+\epsilon))x_3(t) - \gamma(x_1(t_k) + x_2(t_k)) & \end{array}$$

und damit

$$\begin{aligned} x_1(t_{k+1}) &= (1-\gamma(1+\epsilon))x_1(t_{k+1}-1) - \gamma(x_2(t_k) + x_3(t_k)) \\ &= (1-\gamma(1+\epsilon))^2 x_1(t_{k+1}-2) - (\gamma(1-\gamma(1+\epsilon)) + \gamma)(x_2(t_k) + x_3(t_k)) \\ &= \cdots \\ &= (1-\gamma(1+\epsilon))^B x_1(t_k) - \gamma\frac{(1-\gamma(1+\epsilon))^B - 1}{1-\gamma(1+\epsilon) - 1}(x_2(t_k) + x_3(t_k)) \\ &= (1-\gamma(1+\epsilon))^B x_1(t_k) + \frac{(1-\gamma(1+\epsilon))^B - 1}{1+\epsilon}(x_2(t_k) + x_3(t_k)) \end{aligned}$$

Analoge Gleichungen ergeben sich für x_2 und x_3. Damit

$$x(t_{k+1}) = Cx(t_k)$$

$$C = \begin{pmatrix} \alpha & \beta & \beta \\ \beta & \alpha & \beta \\ \beta & \beta & \alpha \end{pmatrix} = (\alpha - \beta)I + \beta ee^T$$

$$\alpha = (1-\gamma(1+\epsilon))^B, \qquad \beta = \frac{(1-\gamma(1+\epsilon))^B - 1}{1+\epsilon}$$

Die Eigenwerte von C sind $\alpha + 2\beta$ und $\alpha - \beta$ mit der Vielfacheit 2 (ee^T hat die Eigenwerte 3 und 0, wobei 0 die Vielfacheit 2 besitzt).

Sei nun γ klein genug gewählt, so daß $\gamma(1+\epsilon) < 1$ gilt. Dann

$$\begin{array}{l} \alpha \to 0 \\ \beta \to -\frac{1}{1+\epsilon} \end{array} \text{ für } B \to \infty \quad \Rightarrow \alpha + 2\beta \to -\frac{2}{1+\epsilon} \Rightarrow |\alpha + 2\beta| > 1 \text{ für große } B$$

Für große B gilt also

$$\rho(C) > 1$$

womit die Iteration nicht konvergiert.

Ist B klein und γ sehr klein, dann gelten die Abschätzungen

$$\alpha = 1 - B\gamma(1+\epsilon) + O(\gamma^2) \approx 1 - B\gamma(1+\epsilon), \qquad \beta \approx -B\gamma$$

Dies ergibt für die Eigenwerte von C

$$\alpha + 2\beta \approx 1 - B\gamma(3+\epsilon), \qquad \alpha - \beta \approx 1 - B\gamma\epsilon$$

Sei γ so klein gewählt, daß $B\gamma(3+\epsilon) < 1$. Dann folgt, daß $\rho(C) < 1$ womit die Iteration für obiges Scenario konvergiert.
Die Konvergenz läßt sich auch noch auf eine andere Weise zeigen, die leichter zu verallgemeinern ist. Dazu betrachten wir für kleine B und γ (wie oben) und für $t_k \leq t < t_{k+1}$

$$\begin{aligned} x_2(t) &= (1-\gamma(1+\epsilon))^{t-t_k} x_2(t_k) + \frac{(1-\gamma(1+\epsilon))^{t-t_k}-1}{1+\epsilon}(x_1(t_k)+x_3(t_k)) \\ &\approx x_2(t_k) - (t-t_k)\gamma((1+\epsilon)x_2(t_k) + x_1(t_k) + x_3(t_k)) \end{aligned}$$

Damit ist $x_2(t) - x_2(t_k) = O(\gamma B)$ und analog $x_3(t) - x_3(t_k) = O(\gamma B)$. Somit unterscheidet sich der Update

$$x_1(t+1) - x_1(t) \;=\; -\gamma((1+\epsilon)x_1(t) + x_2(t_k) + x_3(t_k))$$

vom synchronen Update

$$-\gamma((1+\epsilon)x_1(t) + x_2(t) + x_3(t))$$

durch einen Faktor der Ordnung $O(\gamma^2 B)$. Also haben die veralteten Daten, die bei dem Update von $x_1(t+1)$ verwendet wurden, nur einen Einfluß der Ordnung $O(B\gamma^2)$. Solange also $B\gamma$ wesentlich kleiner als 1 ist, sind die Auswirkungen der Asynchronität gering. Dies heißt, daß der teilweise asynchrone Algorithmus und der synchronen nährungsweise bis zur Ordnung $O(\gamma)$ das selbe Ergebnis liefern. Wir wissen aber, daß der synchrone Algorithmus konvergiert, also konvergiert auch der teilweise asynchrone. Ist $B\gamma$ dagegen größer als 1, dann liegt der 'asynchrone Fehler' in der Größenordnung des Updates selbst und ist damit groß genug um Divergenz zu erzeugen.

- Obige Iteration ist ein Beispiel für einen Algorithmus der teilweise asynchron konvergiert, wenn B klein ist und für große B divergiert.
- Wenn die Verzögerungen klein sind, dann sind die Auswirkungen der Asynchrnonität zu vernachlässigen. Große Verzögerungen haben dagegen eine starken Einfluß auf die Konvergenz. Dies tritt bei Iterationen auf, die eine kleine Schrittweite benötigen.

Satz von Lyapunov

Betrachte die teilweise asynchrone Iteration

$$\begin{aligned} x_i(t+1) &= x_i(t) && \text{für } t \notin T^i \\ x_i(t+1) &= f_i(x_1(\tau_1^i(t)), x_2(\tau_2^i(t)), \ldots, x_n(\tau_n^i(t))) && \text{für } t \in T^i \end{aligned}$$

Aus der teilweisen Asynchronität folgt, daß $x(t+1)$ nur von $x(t), x(t-1), \ldots, x(t-B+1)$ abhängt und nicht von irgendeinem $x(\tau)$, $\tau \leq t-B$. Da Informationen älter als B nicht verwendet werden führen wir den Vektor

$$z(t) = (x(t), x(t-1), \ldots, x(t-B+1))$$

ein. Der Wert von $x(t+1)$ kann nun aus $z(t)$ berechnet werden und damit kann auch $z(t+1)$ aus $z(t)$ ermittelt werden. Induktiv folgt nun daß für jedes Scenario und jede positive ganze Zahl s der Wert von $z(t+s)$ eindeutig durch $z(t)$ bestimmt wird. Ist f stetig, dann ist $z(t+s)$ eine stetige Funktion von $z(t)$ für jedes $s > 0$ bei festem Scenario.
Wir bezeichnen die Menge der Fixpunkte von f mit

$$X^* = \{x \in X : x = f(x)\}$$

Weiter sei

$$Z = X^B$$

und

$$Z^* = \{(x^*, \ldots, x^*) \in Z : x^* \in X\}$$

Für jedes Scenario gilt

$$z(t) = z^* \in Z^* \Rightarrow \forall s > 0 z(t+s) = z^*$$

Satz von Lyapunov. *Wir betrachten eine teilweise asynchrone Iteration. f sei stetig. Weiter existiere eine positive ganze Zahl t^* und eine stetig Funktion $d : Z \to [0, \infty)$ mit den folgenden Eigenschaften*

(a) *Für alle $z(0) \notin Z^*$ und für jedes Scenario gilt*

$$d(z(t^*)) < d(z(0))$$

(b) *Für alle $z(0) \in Z$, für jedes $t \geq 0$ und für jedes Scenario gilt*

$$d(z(t+1)) \leq d(z(t))$$

Dann ist jeder Grenzwert $z^* \in Z$ der Folge $\{z(t)\}$ ein Element von Z^*.

Bemerkungen:

- $d(t)$ mißt den Abstand von $z(t)$ zur Menge Z^*
- $d(z(t))$ fällt monoton
- Obiger Satz ist zu allgemein um sinnvolle Konvergenzraten zu liefern. Man jedoch zeigen, daß für lineare iterative teilweise asynchrone Algorithmen die Konvergenz geometrisch ist.

Beweis. Da f stetig ist folgt, daß $z(t^*)$ eine stetige Funktion von $z(0)$ für jedes Scenario ist. Da auch d stetig ist, ist $d(z(t^*))$ eine stetige Funktion von $z(0)$ für jedes Scenario.
Definiere

$$h : Z \to \mathbb{R}, \qquad h(z(0)) = \max\{d(z(t^*)) - d(z(0))\}$$

wobei daß Maximum über alle möglichen Scenarien angenommen wird. Obwohl es unendlich viele Scenarien gibt, betrachten wir nur ein Zeitintervall der Länge t^*. Damit gibt es nur endliche viele Möglichkeiten, denn wird haben nur eine endliche Zahl von Variablen $\tau_j^i(t), 0 \leq t \leq t^*$ und jede dieser Variablen kan nur B verschiedene Werte annehmen. Also wird das Maximum in der Definition von h über eine endiche Anzahl von Elementen gebildet.
Für jedes Scenario ist $d(z(t^*)) - d(z(0))$ eine stetige Funktion von $z(0)$. Man sieht leicht, daß das

Maximum über eine endliche Zahl stetiger Funktionen wieder stetig ist. Also ist h stetig.
Weiter gilt für jedes Scenario und jedes $z(0) \notin Z^*$

$$d(z(t^*)) - d(z(0)) < 0$$

Also

$$\forall z \notin Z^* \; h(z) < 0$$

Beachte, daß aus der Existenz eines Scenarios, daß mit $z(0) = z$ startet und ein $z(t^*) = \tilde{z}$ erzeugt auch die Existenz eines Scenarios folg, daß mit $z(t) = z$ zur Zeit t startet und $z(t + t^*) = \tilde{z}$ zur Zeit $t + t^*$ erzeugt. Natürlich gilt dies aus umgekehrt.
Deshalb ist obige Funktion h auch gegeben durch

$$h(z(t)) = \max\{d(z(t + t^*)) - d(z(t))\}$$

wobei das Maximum wieder über alle möglichen Scenarien gebildet wird.
Sei nun das Scenario und $z(0)$ fest gewählt. Aus (b) folgt, daß die Folge $\{d(z(t))\}$ nicht wächst. Weiter ist sie nach unten durch 0 beschränkt. Also konvergiert sie gegen ein d^*.
Sei nun $z^* \in Z$ ein Grenzwert von $\{z(t)\}$. Weiter sei $\{t_k\}$ eine Folge, so daß $\{z(t_k)\}$ gegen z^* konvergiert. Dann folgt

$$d^* = \lim_{k\to\infty} d(z(t_k + t^*)) \leq \lim_{k\to\infty} d(z(t_k)) + \lim_{k\to\infty} h(z(t_k)) = d^* + h(z^*)$$

Für alle $z \notin Z^*$ gilt $h(z) < 0$, also folgt $z^* \in Z^*$ ∎

E-mail address: langbein@mathematik.uni-stuttgart.de