文章编号：2096-1618(2016)06-0575-08

回复式神经网络优化逆运动问题求解

金 虎[1]， 陈 超[2]， 陈念伟[1]
(1. 成都信息工程大学信息安全学院，四川 成都 610225；2. 96313 部队，湖南 邵阳 422000)

摘要：针对传统逆运动问题数值求解算法计算时间长，多解情况下计算结果单调的问题，采用回复式神经网络对逆运动问题进行近似优化求解。通过针对逆运动问题建立基于回复式神经网络的动态求解模型，采用神经元异步更新计算和初始状态扰动，保证多解情况下解运动迹线的多样性。文中算法在无解情况下有较好适应性和稳定性，能迅速收敛到近似最优状态。算法理论计算时间复杂度为 $O(n^2)$，可满足实时应用的需求。实验对典型对子运动链运动系统进行模拟，结果表明算法具有稳定和收敛性。

关 键 词：计算智能；逆运动；回复式神经网络；运动曲线；异步计算
中图分类号：TP183 **文献标志码**：A

0 引言

研究采用回复式神经网络优化模型求解逆运动问题。回复式神经网络求解模型可避免逆运动系统随自由度增加计算量呈指数增长问题，使计算时间具有稳定性，可有效控制三维运动仿真的数值计算开销。此外，传统确定性迭代算法如循环坐标速降(cyclic coordinate descent，CCD)在对具有相似初始状态和终止状态的多解问题求解时，计算得到运动轨迹存在单一性。采用神经网络近似优化求解方法可避免上述问题，当多解存在时，通过初始状态的扰动易于获得多样性的解，进而得到不同的运动迹线。

逆运动是机器运动学中的概念，是指使用运动学方程求解机器人关节运动的轨迹，最终使得机器人的终端执行器达到既定的位置[1-2]。逆运动问题的求解广泛存在于当前应用领域中。除机器人运动学外，在交互式计算机图形学和计算机三维游戏设计、虚拟现实等领域中的应用也日益广泛[3]。

逆运动问题的计算方法可大致划分为：解析计算方法、数值计算方法和优化求解方法3种类型。(1)解析计算方法在实际的逆运动问题求解中应用相对较少，主要通过对关节链的运动描述，建立运动方程，针对运动方程进行解析运算的求解方式[4-6]。在多数情况下，这种方法求运动方程解析解的过程较为复杂(例如6-R序列链)[4]，且求解计算难以实施，所以该方法仅适用于部分特定的应用场景。(2)逆运动问题的数值求解方法又可大致划分为两大类[7-8]。一类采用Newton-Raphson数值方法求解非线性逆运动方程[9-11]。在这类研究中，逆运动方程被转换为带约束的非线性编程问题，采用quasi-Newton及改进算法求解。另一类则采用预测-校正算法求解运动微分方程。数值计算方法的主要困难在于特定情况下的Jacobian矩阵为奇异时，无法直接计算运动方程的解。此外，当解向量初值不准确时，求解的计算过程可能会不稳定。针对Jacobian矩阵为奇异情况，可采用Jacobian伪逆矩阵或Jacobian扩展逆矩阵[11]进行求解。对于重复运动或周期性的应用场景，奇异性即表明多解存在，此时采用数值计算从相同或相似的初值开始运算，所获得的都是相同单一解，难以体现出多值解的多样性，会导致动画结果表现单调[12]。(3)近年有较多的研究热衷于对逆运动问题运用优化求解方法[7,11]。在Cartesian坐标系下，逆运动系统的终端执行器当前状态和目标状态可用位置、方向向量表示。终端执行器与目标之间的位置和方向偏差为计算目标。在文献[13-16]中，采用l-正则化损失最小问题来概化该类问题，并采用CCD算法求解。算法每轮计算仅允许一个关节运动，单关节运动时以最贪婪方式趋近于目标。算法具有简单快速的优点，但对于带约束问题，约束不满足时的回溯会极大增加计算量，且易导致求解过程的不稳定。

采用神经网络的优化求解能力来解决逆运动问题。与上述多层神经网络的强化学习模式不同，文中采用近似优化方法，并设计出针对运动链的计算模型进行求解。逆运动系统的目标与终端执行器位置差异为优化目标，用神经网络节点表示系统运动状态，通过寻求系统能量最小的稳定点完成目标函数的求解过程。基本工作包括：(1)逆运动问题的优化求解表达；(2)回复式神经网络求解逆运动问题求解模型；(3)带

收稿日期：2016-10-20
基金项目：四川省科技厅科技支撑计划资助项目(2013GZX0169)

约束目标优化问题的回复式神经网络求解算法;(4)与 CCD 算法的对比试验方案设计和实现。设计的回复式神经网络求解逆运动算法的主要特点包括:(1)神经元异步更新计算;(2)约束满足求解;(3)多解条件下可方便得到多样性的运动迹线。

1 逆运动神经网络模型

1.1 逆运动问题

逆运动在刚体(rigid body)约束系统的运动分析中是典型示例。在 Euclidean 空间中,粒子运动是其在惯性 Cartesian 坐标系下每一时刻的位置描述。在三维正交轴坐标系中,用三元组 $(x,y,z) \in R^3$ 标记其位置,每组坐标值为位置在对应轴上的投影。动态系统中,运动迹线为时间参数的曲线方程 $p(t)=(x(t),y(t),z(t)) \in R^3$。逆运动系统更关注集合对象的运动状态而非单个粒子轨迹。集合对象之间通过连接杆结合在一起,即通常表述的刚体运动系统。设 p,q 为刚体系统中任意两个粒子对象,则刚体系统可 $||p(t)-q(t)||=||p(0)-q(0)||=C$ 表示。即在运动过程中,严格保持粒子对象之间的距离不随时间改变[1]。

刚体系统中的节点与连接的运动通常不会是完全的自由运动,相互之间存在制约或关联。表征刚体系统运动状态的一个主要特征称为系统自由度个数(DOF)[14],它等价于系统中独立运动参数的个数。自由度是系统运动分析的基础,自由度的个数决定系统运动特征也决定了问题的计算规模。在运动方程表达中,用自由度表征函数输入变量个数,则运动函数为映射函数,用于将自由变量映射到系统的运动输出空间,即决定了系统运动的最终位置[15-16]。

$$\dot{\theta} = G(\theta)u = \sum_{i=1}^{m} g(\theta)u_i$$
$$y = k(\theta) \tag{1}$$

其中,$\theta \in R^n, y \in R^r$,分别为刚体运动空间和目标空间。$u \in R^m$ 为运动控制参数。若控制函数在连续时间间隔$[0, T]$区间可满足映射函数空间 χ,则系统运动为控制系统对初始格局向终端目标的映射[11]。r 为运动目标的维度,n 为可运动变量的维度,p 为运动系统的自由度,其中 $n \geqslant p \geqslant r$。通常运动变量维度 n 并非最简系统自由度 p。

运动系统中的自由度决定了逆运动问题求解的计算规模。将高维度的自由度空间向低维度的目标空间投影,且映射为非线性。逆运动求解形式非常适合于含隐层神经元的神经网络模型,这也是大多强化学习网络模型常用求解方式。

1.2 运动链求解优化

刚体运动系统中,连接器和关节间形成运动对。多个连接器与关节链接形成不同的运动链形式,又称这种运动链为对子(dyad)链接。对子链接一般可划分为几种基本类型:旋转-旋转-旋转(RRR),旋转-旋转-平移(RRT),旋转-平移-旋转(RTR),平移-旋转-平移(TRT),平移-平移-旋转(RTT)。复杂的运动系统均可分解为上述基本类型的对子链接的组合。因此,对该类型问题的求解有普适性。

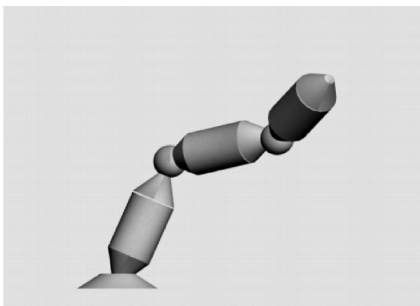


图 1 RRR 类型对子链

在运动链中,有一相对不动的基点称为静地点,如图 2 中"O"点。一些研究中,将静地点的坐标系作为固定参照系,称为主参照帧。运动链上其他关节的运动为相对帧。按照相对运动顺序,可把运动链上关节按层次表示,例如 Hodgins 的机器人运动模型,按照树形结构表示躯干与四肢之间的关系。

$$\text{ChildWorldCoord} = \text{ParentWorld} \times \text{Local}$$

CCD 算法[19]中,则采用了层状迭代方式更新关节状态,但这种表示并不适合异步更新的处理方式。

求解 IK 问题,即连接-关节的格局,需满足终端执行器与目标的一致。刚体系统执行器 $s \in R^r$,且 s 包含于 θ。分别用列向量 $\vec{s} = (s_1, s_2, \ldots, s_r)^{\mathrm{T}}$ 和 $\vec{y} = (y_1, y_2, \ldots, y_r)^{\mathrm{T}}$ 表示执行器和目标状态,二者之间的差异用误差表示:$\vec{e} = \vec{y} - \vec{s}$。系统运动状态的格局由关节点列向量表示 $\vec{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)^{\mathrm{T}}$。执行器状态 s 受控于关节点,为关于 θ 的映射:$s_i = s_i(\theta), i = 1, 2, \ldots, r$。若逆运动存在有效解时为 $y_i = s_i(\theta)$。则逆运动求解等价于误差向量 $\vec{e}$ 关于 θ 的优化问题,用一阶正则优化问题(l_1-regularized loss minimization problems)[7]表示:

$$\frac{1}{m}\sum_{i=1}^{m} l(\theta, Z_i) + \lambda ||\theta||_1 \tag{2}$$

$Z_i = (s_i, y_i)$ 表示执行器格局与目标点对的关系函数,θ 为高维关节点在 Z 状态下的格局,l 则为凸损失函数(convex loss function)表征学习性能,$\lambda \geq 0$ 为

正则参数。一阶范数 $\|\theta\|_1$ 作为正则项或惩罚因子,用于调整学习速率。

实际计算中,(2)式可进一步抽象为凸优化问题的特例情况:

$$\min_{\theta\in R^n} F(\theta) := f(\theta) + \lambda \|\theta\|_1$$

显式的优化处理过程为计算优化决策点,即求解 $\theta^* \in \theta$,可满足:

$$F(\theta^*) = \min\{F(\theta): \theta \in R^n\}$$

逆运动系统除了使终端执行器满足到达目标要求之外,通常会存在对关节点和连接器的运动存在约束的情况。逆运动转换的凸优化极值问题,可由下式的带约束优化形式表示:

$$\begin{aligned} &\text{minimize} && F(\theta)\\ &\text{subject to} && g_i(\theta) \le 0, \quad i = 1,2,\ldots,m\\ &&& h_i(\theta) = 0, \quad i = 1,2,\ldots,p \end{aligned}$$

不等式约束形式的 $\geqslant$, $\leqslant$ 是对偶形式,统一由 $g(\theta) \leqslant 0$ 形式表示。等式约束 $h=0$ 可等效替换不等式约束 $h\leqslant 0$ 和 $-h(x) \leqslant 0$。

根据 $F(\theta)$ 函数为 θ 变量的线性表示时,极值优化为 $\min\{c^T\theta: \theta \in R^n\}$,该问题为典型线性编程问题。当目标函数 $F(\theta)$ 是关于 θ 变量的非线性函数时,该问题则是典型的非线性编程问题。在非线性编程问题中,二次型优化是重要的非线性编程形式:

$$F(\theta) = \frac{1}{2}\theta^{\mathrm{T}} Q\theta + C^{\mathrm{T}}\theta$$

其中,Q 为对称矩阵形式。在逆运动问题中,终端执行器和目标向量之间的误差优化可表现成二次型优化问题。

把逆运动问题转化成优化问题,并通过典型 BP 算法求解是常见的 IK 求解方法。但通过输入-输出数据对的训练网络方式,对于运动系统状态方程确定的情况,没有充分利用已知的系统状态信息。回复式神经网络模型则可充分利用已有信息,实现更好的求解方法。

2　回复式神经网络求解模型

2.1　回复式神经网络模型

人工神经网络模型具有很强的非线性近似能力[17-19],是非线性编程问题的典型求解模型。人工神经网络可根据信号的传输模式划分为:前向式神经网络和回复式神经网络两种模型。前向式神经网络模型也称为多层感知器模型,模型作为通用的近似函数得到了深入的研究,其中以 BP 神经网络模型为代表,通常使用生成-检验的学习方法,广泛地应用于曲线拟合,模式分类,非线性系统认知等。回复式神经网络的基础是动力学模型,单元之间的连接形成有向环,这也使网络内部状态展示出动态的时间特性。

神经网络由神经元和网络连接构成。神经元为基本计算单元,神经网络的计算能力主要决定于网络的连接,知识存储在网络的连接上。图 2 为典型的回复式神经网络示意图。

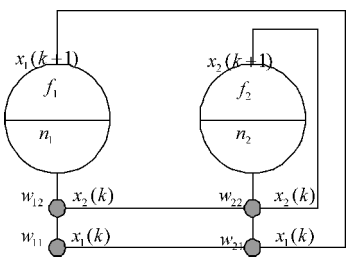


图 2　双神经元回复式全连接图

在单层回复式网络中,所有节点都构造一样,节点之间互相连接,既可接受其它节点的输入,也可输出至其它节点。节点间对输出反馈的引入,使网络成为一个非线性动力系统。

$$\begin{cases} x_1(k+1) = f_1(w_{11}x_1(k) + w_{12}x_2(k)) \\ x_2(k+1) = f_2(w_{21}x_1(k) + w_{22}x_2(k)) \end{cases} \tag{3}$$

将规模扩展到同型多神经元网络,可用更一般形式表示为

$$x_i(k+1) = f_i\left(\sum_{j=1}^{n} w_{ij}x_j(k)\right) \tag{4}$$

其向量形式为

$$x(k+1) = f(wx(k) + b)$$

等式两方同时加上 $-x(t)$,得

$$x(k+1) - x(k) = -x(k) + f(wx(k) + b)$$

作连续化变换,可得动力方程形式:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -x(t) + f(wx(t) + b) \tag{5}$$

根据 f 函数的表现形式,可将动力系统划分为线性或非线性情况。当 $f(s) = s$ 即为典型的线性系统。常见 f 函数的形式有:

阈值函数型:

$$\varphi(v) = \begin{cases} 1, & v \ge 0 \\ 0, & v < 0 \end{cases}$$

也称为 M-P 神经元模型。

正弦双曲正切函数:

$$\varphi(v) = \tanh(v/2) = \frac{1 - \exp(-v)}{1 + \exp(-v)}$$

Sigmoid 函数类型:

$$\varphi(v)=\frac{1}{1+\exp(-\alpha v)}$$

Sigmod 函数具有平滑渐进,以及可保持单调性优点,参数 α 控制平滑度。

对于线性系统:$\dot{x}=wx$ 从任何初始点位置出发都会有唯一的解,$x(t)=e^{wt}x_0$ 对所有 $t\geq 0$。但对于非线性系统 $\dot{x}=f(x)$ 来说,并不存在这样理想的结果,只有少数特殊情况有可能求解非线性系统方程。虽然难以直接对非线性系统方程进行求解,但大量研究表明,可以通过解的局部行为实现对等价非线性系统的分析。矩阵 $\boldsymbol{A}=Df(x_0)$ 为 f 函数在 x_0 出的导函数,矩阵 $\boldsymbol{A}$ 也称为非线性系统的 Jacobian 矩阵。Hartman-Grobman 定理表明,可通过检测平衡点 x_0 的 $Df(x_0)$ 矩阵的特征值 λ 判断其稳定性。在非线性系统的分析中,Lyapunov 定理提供相对简单的稳定计算方法[13]。

定义:考虑动力系统的一般方程 $\mathrm{d}x/\mathrm{d}t=f(t,x)$,其中函数 $f(t,x)$ 对 $x\in G\subset R^n$ 和 $t\in(-\infty,\infty)$ 连续,并对 x 满足李氏(Lipschitz)条件。若对任意给定的 $\varepsilon>0$,都存在 $\delta=\delta(\varepsilon)>0$,只要:

$$|x_0-\varphi(t_0)|<\delta$$

方程以 $x(t_0)=x_0$ 为初值的解 $x(t,t_0,x_0)$ 在 $t\geqslant t_0$ 有定义,且满足:

$|x(t,t_0,x_0)-\varphi(t)|<\varepsilon$, 对所有 $t\geq t_0$,

则称解方程(动力系统)的解 $x=\varphi(t)$ 是李雅普诺夫意义下稳定。在解稳定条件下,若存在 $\delta_1(0<\delta_1<\delta)$,可满足:

$$|x_0-\varphi(t_0)|<\delta_1$$

即有:$\lim\limits_{t\to+\infty}(x(t,t_0,x_0)-\varphi(t))=0$。

则称解 $x=\varphi(t)$ 是李雅普洛夫渐进稳定的。若解 $x=\varphi(t)$ 不是稳定的,则称为不稳定。

Lyapunov 定理 假定存在一个实值函数 $V(x)$ 满足 $V(x_0)=0$,且当 $x\neq x_0$ 时,$V(x)>0$,则:

(1)若对所有 x, $\dot{V}(x)\leq 0$,则 x_0 稳定;

(2)若对所有 $x\neq x_0$, $\dot{V}(x)<0$,x_0 为渐进稳定;

(3)若对所有 x, $\dot{V}(x)>0$, x_0 不稳定。

函数 $V:R^n\to R$ 满足上述定理则称为 Lyapunov 函数。Lyapunov 定理通过构造 Lyapunov 函数 $V(x)$ 来确定系统的稳定性。Lyapunov 定理提供了有效的方法用于求解神经元动力系统。与 Lyapunov 方法类似的还有能量函数方法。

2.2 逆运动问题的回复式神经网络模型

在逆运动问题中,求解的优化形式按式(2)表示,该表示形式有通用性。考虑仅有位置要求的优化问题,执行器与目标位置重合。在 Cartesian 空间下,可用回复式神经网络模型实现该问题的优化求解。刚体系统中,根节点固定,关节点 θ_i 的运动形式为 R^3 旋转,关节点之间的连接不变形。刚体的旋转和滑动可用统一的变换矩阵表示。

刚体系统的逆运动目标函数采用欧拉距离表示为

$$\begin{aligned}&f(x_1,x_2,x_3,\cdots,x_{3i-2},x_{3i-1},x_{3i},\cdots,x_{3n-2},x_{3n-1},x_{3n})\\&=(y_3-x_{3n})^2+(y_2-x_{3n-1})^2+(y_1-x_{3n-2})^2\\&\text{minimize } f(x_1,x_2,x_3,\cdots,x_{3n-2},x_{3n-1},x_{3n})\end{aligned}$$

其中,对每一中间关节点 θ_i ,用神经元位置 $3i-2$, $3i-1$, $3i$ 表示对应 Cartesian 的 R^3 座标。

关节点满足刚性系统中连接器不变形的约束条件。

$$g_i=|\theta_i-\theta_{i-1}|=\rho_i \tag{6}$$

$$(x_{3i}-x_{3(i-1)})^2+(x_{3i-1}-x_{3(i-1)-1})^2+(x_{3i-2}-x_{3(i-1)-2})^2=\rho_i^2$$

i 从 2 到 n,约束产生在邻近的两个关节节点之间。如果关节点的旋转角存在其他约束,则只是增加约束函数的数量,对问题求解并无本质影响。

由式(6)可见,约束函数 j 仅与当前关节点与相邻的前一关节点位置相关,因此对这该二次型的导数计算易于生成统一表达式。

对单个神经元的计算可使用 canonical 非线性编程动态模型表示:

$$C_i\frac{\mathrm{d}x_i}{\mathrm{d}t}=-\frac{\partial f}{\partial x_i}-\sum_{j=1}^{q}\mu_j\frac{\partial g_j}{\partial x_i} \tag{7}$$

其中,g_j 为约束条件所对应的函数,$j=1,2,\cdots,n$,为约束函数的个数。$i=1,2,\cdots,3n$。i_j 为非线性激励形式 $\mu_j=\varphi_j(g_j(x))$,激励函数 $\varphi_j(x)$,$x\in R$ 可选如下用有上饱和阈值的形式表示。

$$\varphi_j(x)=\begin{cases}0 & x>0\\ x & x\leqslant 0\end{cases}$$

该函数可满足 $x\times\varphi_j(x)\geq 0$。

上述优化模型中,函数 $\varphi_j(x)$ 满足连续条件,且目标函数 $f(x)$ 与约束函数 $g(x)$ 及其导数均满足连续条件,可设定 Lyapunov 函数为

$$E(x)=f(x)+\sum_{j=1}^{q}\int_0^{g_j(x)}\varphi_j(\theta)\mathrm{d}\theta \tag{8}$$

对该函数求导数,按照 Lyapunov 定理,计算 $\dot{E}(x)$ 的值,确定该优化问题是否存在稳定解。

$$\frac{\mathrm{d}E}{\mathrm{d}t}=\sum_{i=1}^{n}\frac{\partial f}{\partial x_i}\cdot\frac{\mathrm{d}x_i}{\mathrm{d}t}+\sum_{j=1}^{q}\sum_{i=1}^{n}\varphi_j(g_j(x))\frac{\partial g_j}{\partial x_j}\cdot\frac{\mathrm{d}x_i}{\mathrm{d}t}$$

其中,$\partial f/\partial x_i$ 由式(7)替代,

则第一项 $\sum\limits_{i=1}^{n}\frac{\partial f}{\partial x_i}\cdot\frac{\mathrm{d}x_i}{\mathrm{d}t}=-\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{q}\mu_j\frac{\partial g_j}{\partial x_j}\cdot\frac{\mathrm{d}x_i}{\mathrm{d}t}-$

$$\sum_{i=1}^{n} C_i \frac{\mathrm{d}x_i}{\mathrm{d}t} \cdot \frac{\mathrm{d}x_i}{\mathrm{d}t}$$

即 $\frac{\mathrm{d}E}{\mathrm{d}t} = -\sum_{i=1}^{n} C_i \frac{\mathrm{d}x_i}{\mathrm{d}t} \cdot \frac{\mathrm{d}x_i}{\mathrm{d}t} \leq 0$

可见,对上述二次型优化的回复式神经网络模型,在保证控制参数 C_i 值非负即可保证(8)式满足 Lyapunov 的稳定条件,系统存在稳定点 x_s。

在上述回复式神经网络计算模型的基础上,给出 IK 系统求解算法步骤如下:

(1)确定 IK 系统的输入向量,初始化 IK 系统初态。

$$x^0 = (x_1, x_2, \cdots, x_p)^0$$

(2)设置逆运动的目标函数,表达式为运动目标值与系统执行器的距离平方和。

$$E_p(t) = \frac{1}{2} \| d_p - y_p(t) \|_3^2 = \frac{1}{2} \sum_k [d_{kp} - y_{kp}(t)]^2$$

$$= \frac{1}{2} \sum_k e_{kp}^2(t)$$

(3) 设定约束函数表达式 $g_i(x)$,对 IK 系统应用刚体对象不变形约束条件。

$$g_i(x) = \| v_i - v_{i-1} \|^2 - R_i^2 = 0$$

v_i 为关节连接点 i 的位置坐标

(4)按随机排列 order = Permu(n)的顺序,选择关节变量 i,结合式(5)和式(7)更新 IK 状态。

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = \left(-\frac{\partial f}{\partial x_i} - \sum_{j=1}^{q} \mu_j \frac{\partial g_j}{\partial x_i} \right) / C_i$$

C_i 为针对关节点 i 的更新常系数,需根据实际 IK 问题进行设定调整。

$$x^{k+1} = h(x^k) = (y_1, y_2, \ldots, y_s), k \to +\infty$$

向量上标表示运算过程中向量的迭代次数

(5)重复步骤(4),计算当 k 到达设定的设定的迭代次数 niteration 或 delta(x^{k+1}, x^k) < $c_{threshold}$ 时停止。

算法中单个神经元计算时间由第(4)步决定,取决于约束函数的个数以及相关其他神经元的交互作用两个方面。对单个神经元更新的计算时间应小于 $k \times n$。其中,k 为与约束个数相对应的常数,n 为神经元的规模。所以,该步计算时间复杂度应为 $O(n^2)$ 规模。

3　试验方法和试验方案

3.1　解的收敛性与多样性

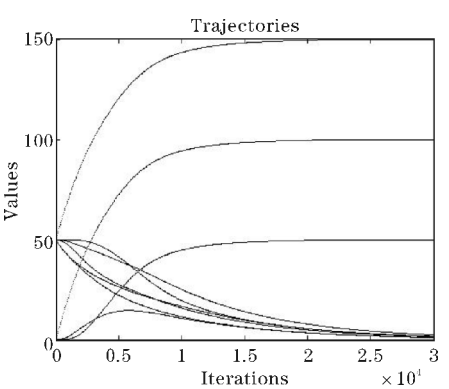
试验环境为 Windows XP 操作系统,软件系统为 Matlab-R 2010b。在该环境下,首先设计试验测试回复式神经网络方法对逆运动问题的求解能力。在一多关节的逆运动系统中,按 Cartesian 坐标系设定连接和关节的初始状态,分别根据问题无解、有唯一解或多解情况设置系统的运动目标。按典型的对子链接为 RRR 类型时,试验回复式神经网络的求解能力。其次,设计与 CCD 算法的对比实验,分析回复式神经网络模型求解算法的优缺点。

表 1 为三节点的链接系统。其中链接长度 $R_1 = R_2 = R_3 = 50$,初始状态为[0 50 0 50 50 0 50 50 50],控制参数 C 设定为 10000,神经元个数为 9,设定运算迭代次数为 300000。

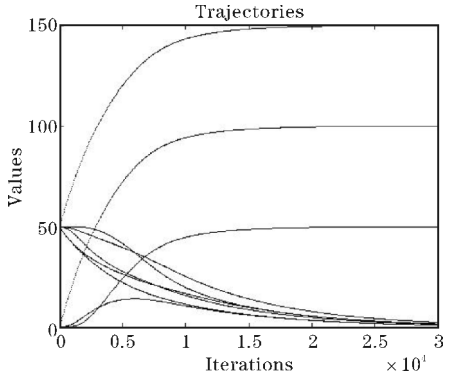
表 1　三节点 RRR 类型求解表

解类型	解状态	
	目标状态	最终状态
无解	(0 0 200)	(0.00 0.00 50.01 0.00 100.20 0.00 0.00 150.03)
唯一解	(0 0 150)	(1.64 3.28 49.87 1.65 3.29 99.87 0.01 0.02 149.73)
多解	(0 0 50)	1. (-6.73 49.52 -1.46 30.94 30.70 25.50 0.00 0.00 50.0)
		2. (-6.71 49.53 -1.50 30.95 30.65 25.45 0.00 0.00 50.0)
		3. (-6.63 49.54 -1.50 30.97 30.69 25.53 0.00 0.00 50.0)
		4. (-6.61 49.54 -1.50 30.98 30.69 25.54 0.00 0.00 50.0)
		5. (-6.72 49.52 -1.47 30.95 30.66 25.46 0.00 0.00 50.0)

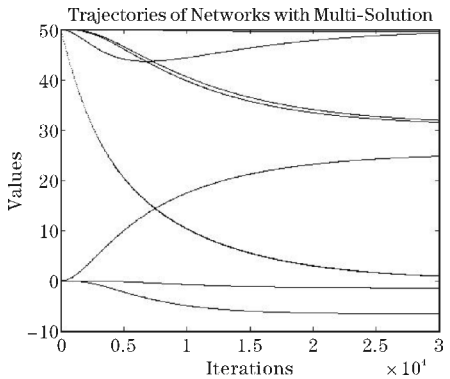
表 1 中数据为方便阅读,数值精度仅保留了小数点后两位。从表 1 可见,回复式神经网络对无解、唯一解或多解情况都有很好的收敛性。在无解情况下,终端执行器可获得满足约束条件下最接近目标点的状态。表 1 中无解状态对应于目标对象不可达情况,神经网络状态迹线变化如图 3(a)所示。计算在迭代次数约束下停止,此时系统状态处于约束所条件下的最优状态,即执行器所能达到距离目标最近的位置。图 3(b)对应表 1 中有唯一解时的迹线变化过程,计算收敛并得到唯一最优解。图 3(c)为多解求解过程中的一次求解过程的迹线变化过程。



(a)无解条件的网络状态迹线



(b) 单解条件的网络状态迹线



(c) 多解条件的网络状态迹线

图 3 神经网络的状态迹线

图 3 采用固定迭代计算次数来分析迹线变化状况。整体上前期计算速度较为迅速,很快进入平滑收敛区域。3(c)所示多解状态的计算具有最好的收敛性,无解状态计算收敛速度略好于单解状态的计算速度。

对多解情况,算法在上述参数设定下并未体现出太大的多样性,基本收敛于单一的终止状态。为使多解条件下的运算呈现多样性的特点,从初始状态与控制参数的调整两个方面进行试验。

针对上述目标状态为(0 0 50)的多解情况,保持系统初始状态,分别设置 $C=1000,2500,5000,8000$,获得计算终止状态如表 2 所示。

表 2 控制参数设置与解的多样性

控制参数	最终解状态
$C=1000$	(--- --- --- --- --- --- --- ---)
$C=2500$	(--- --- --- --- --- --- --- ---)
$C=5000$	1. (−16.9978 47.0217 −0.1876 24.7825 35.2403 24.6239 0 0 50) 2. (−10.6200 48.8295 1.7020 29.9335 31.4922 25.2568 0 0 50) 3. (5.8111 49.3830 −5.2492 33.8887 32.2919 32.4277 0 0 50)
$C=8000$	1. (−3.6084 49.8322 −1.9326 32.3507 30.1955 26.7264 0 0 50) 2. (−7.1976 49.4557 −1.5259 30.6574 30.8285 25.3074 0 0 50) 3. (−6.8923 49.4986 −1.5434 30.7989 30.8389 25.4971 0 0 50)

在控制参数取值较小时,回复式神经网络算法未能计算收敛求得有效解。当控制参数值高于一定阈值时,解的多样性随控制参数值增高而呈下降趋势。由分析可知,随控制参数值增大,算法退化成数值计算方法,计算结果趋于单一性。因此,适当调整控制参数的选取范围,可有效提高解的多样性。

同样,保持控制参数值不变条件下,通过对初始状态的微小扰动,考查初态波动下的解多样性状况。在 $C=10000$,最大迭代次数为 30000,得到表 3。

表 3 初值与解多样性

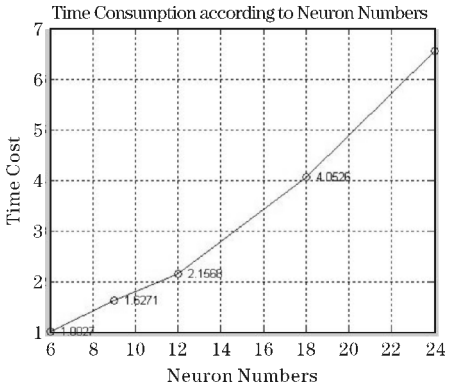
	数据序列					
Neuro	$State_0+\delta_1$	Final state1	$State_0+\delta_2$	Final state2	$State0+\delta_3$	Final state3
1	4.47	−5.8722	0.00	−3.7879	0	−0.1232
2	49.8	49.6296	50.00	49.8250	49.8	49.7325
3	0	−1.5544	0.00	−1.7677	4.47	5.1636
4	50.47	31.3742	49.80	32.3763	0	−12.4961
5	49.8	30.4424	45.5323	29.9844	49.8	48.2803
6	0	25.7323	0.00	26.4901	50.47	53.5868
7	50.47	0.0000	49.80	0.0000	50	0.0000
8	49.8	0.0000	45.5323	0.0000	49.8	0.0000
9	50	50.0000	50.00	50.0000	50.47	50.0000

表 3 数据表明,对初始状态的微小扰动,也可达到求解结果多样性的效果。因此,在实际应用中可通过综合使用控制参数值与初始状态扰动方式来获得多样性的解。

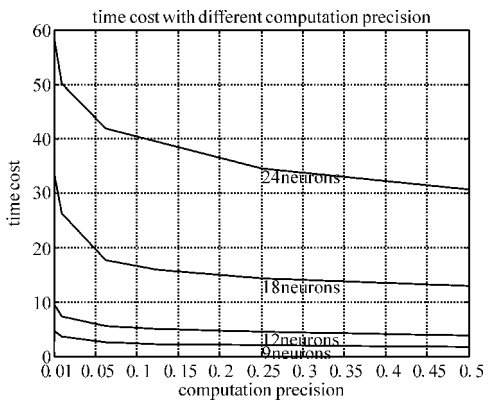
3.2 计算性能分析

在满足解的收敛性和多样性情况下,分析算法的计算性能。设定最大迭代次数为 30000,按关节点个数分别为 2,3,4,6,8 条件下,算法的平均运行时间,绘

制出神经元个数与运行时间之间的关系如图 4 所示。



(a) 求解时间与神经元数量的关系



(b) 不同神经元数量在不同精度下的求解时间

图 4　回复式神经网络求解的时间性能

图 4(a)为固定最大迭代次数为 30000 时,使用不同自由关节个数系统的计算时间花销。由上图可看出,随系统中神经元个数的增长,系统花销也呈正变增长,但并非是指数增长关系。这也可从算法的理论计算时间分析上得到验证。对单个神经元的更新依赖于系统中其他神经元的作用,在所有神经元的一轮更新中,计算开销为 $O(n^2)$强度。因此,对于有实时时间需求的计算,该方法的时间开销是可接受的。图 4(b)为不同计算精度下的实验结果,可见在降低近似计算精度要求的情况下,算法可以快速获得计算结果。

4　结论

通过设计回复式神经网络模型优化逆运动问题求解,在求解速度、稳定性和多样性方面都体现出良好的性能,为解决带约束的运动系统分析提供了一个有效的方法。所设计模型在目标可达性方面体现出较好的适应性,可稳定收敛于系统所能达到的最佳位置。这种适应性与动力系统函数表达,以及模型结构之间的本质联系是今后进一步研究的方向。

参考文献:

[1] Richard P Paul. Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators[M]. Massachusetts. MIT Press, Cambridge, 1981:65-85.

[2] Richard M Murray. Mathematical introduction to Robotic manipulation [M]. CRC Press, 1994:81-146.

[3] Rick Parent. Computer Animation: Algorithms and Techniques[M]. MORGAN KAUFMANN Publishers, 2002:175-216.

[4] AJ Turner, JF Miller. Recurrent Cartesian Genetic Programming Applied to Series Forecasting[M]. Springer International Publishing, 2014:476-486.

[5] Oleg Ivlev, Axel Gräser. An Analytical Method for the Inverse Kinematics of Redundant Robots[C]. in Proc. 3rd ECPD Int. Conf. on Advanced Robots, Intelligent Automation and Active Systems 1997:416-421.

[6] Deepak Tolani, Ambarish Goswami, Norma I Badler. Real-time inverse kinematics techniques for anthropomorphic limbs [J]. Graphical Models 2000, 62:353-388.

[7] L Zhang, G Brunnett. Combining inverse blending and Jacobian-based inverse kinematics to improve accuracy in human motion generation[J]. Computer Animation and Virtual Worlds, 2016, 27(1):3-13.

[8] I-Ming Chen, Guilin Yang. Inverse Kinematics for Modular Reconfigurable Robots[C]. Proceedings of the 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium, May, 1998:1647-1652.

[9] N I Badler, D Tolani. Real-Time Inverse Kinematics of the Human Arm[J]. Presence, 1996, 5(4): 393-401.

[10] Gaurav Tevatia, Stefan Schaal. Inverse Kinematics for Humanoid Robots[C]. In Proceedings of the International Conference on Robotics and Automation (ICRA2000), 2000:294-299.

[11] KRZYSZTOF TCHON, JOANNA KARPINSKA, MARIUSZ JANIAK. Approximation of Jacobian Inverse Kinematics Algorithms[J]. Int. J. Appl. Math. Comput. Sci., 2009, 19(4):519-531.

[12] Radek Grzeszczuk. NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models[C]. thesis for degree of Doctor of Philosophy, 1998:26-43.

[13] H J Tang, K C Tan, Zhang Yi. Neural Networks: Computational Models and Applications [M]. Springer-Verlag, 2007:57-79.

[14] Dan B. Marghitu, Mechanisms and Robots Analysis with MATLAB[M]. New York: Springer Dordrecht Heidelberg London, 2009:1-10.

[15] Samuel R Buss. Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods, ucsd. edu/sbuss/ResearchWeb, 2004:1-19.

[16] P TSENG. Convergence of a Block Coordinate Descent Method for Non-differentiable Minimization [J]. Journal of Optimization Theory and Applications, 2001, 109(3):475-494.

[17] Mustafa Ayyıldız, Z Mustafa, K Etinkaya. Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-DOF serial robot manipulator[J]. Neural Computing and Applications archive, 2016, 27(4): 825-836.

[18] Eimei Oyama, Arvin Agah. A modular neural network architecture for inverse kinematics model learning[J], Neuro-computing, 2001, 38(40): 797-805.

[19] M Z Al-Faiz, MIEEE. Inverse Kinematics Solution for Robot Manipulator Based on Neural Network[J]. MASAUM Journal of Basic and Applied Sciences, 2009, (2):147-154.

Optimal Inverse Kinematics Solution using Recurrent Neuron-Networks

JIN Hu[1], CHEN Chao[2], CHEN Nian-wei[1]

(1. College of Information Security Engineering, Chengdu University of Technology and Information, Chengdu 610225, China; 2. 96313 unit, Shaoyang 422000, China)

Abstract: The conventional Inverse Kinematics problem solving methods have many disadvantages such as slowly convergence speed or monotonous computation result. This issue presents a novel algorithm with recurrent neuron-networks model to treat the Inverse Kinematics problem as approximate optimization solving. This method designs proper optimization function for Inverse Kinematics system and utilizes recurrent neuron-networks computational model to resolve. During the computation, the neurons were updated asynchronously and the initial system state was disturbed to obtain diverse trajectories when multi-solutions exist. The algorithm has good stability and convergence ability even when no solution exists. The theoretical computation time complexity of the algorithm is $O(n^2)$, and can meet general real-time application requirements. Experiments were done to simulate the dyad kinematics system. The calculation results show that the computational time are increase with the neuron number but not exponentially, that are in good agreement with the algorithm convergence stability.

Key words: computational intelligence; Inverse Kinematics system; recurrent neural networks; trajectory planning; asynchronous calculation