群知能を用いたロボットの戦略的制御方法の検討

○猪原　知俊*1　入江　寿弘*2　新宮　清志*3

キーワード：戦略的制御　多目的最適化　群知能　シミュレーション　2 脚ロボット

１．はじめに

３次元空間を移動可能な自律移動ロボットは、様々な環境情報を元に与えられた目的にしたがって、その行動計画をロボット自身が作成・行動する高度な制御が必要となる。本稿では、脚機構ロボット[1]における歩行計画の手法を例に、群知能アルゴリズムを用いた「戦略的制御」の方法について検討する。

２．戦略的制御

最適化手法の適用分野の１つとして多目的最適化が注目されているが、最適化目的が複数あるとしても何か決まった評価関数（あるいは適応度関数）の最適値を求め、その最適値周辺の類似の解を広く複数求めることで結果的にそれを多目的な解と捉えている場合がある。本研究で提案する方法では、複数の目的関数を一つの評価関数にまとめる際に、敢えて許容値を持つ目的関数を設定することで幅を持たせた解を探索するように試みる。このような評価関数を導入することにより線形に評価する通常の最適化手法では得られないような特異な解が得られる可能性が考えられる。この手法を用いる制御を「戦略的制御」と呼称することとする。

３．歩行方法

以下に歩行パターン動作①～⑥を示す。

① 重心を右に移動し、左足を持ち上げ半歩前に出す
② 左足を更に半歩前に出す
③ 右足を半歩後ろに下げ、左足を着地させる
④ 重心を左に移動し、右足を持ち上げ半歩前に出す
⑤ 右足を更に半歩前に出す
⑥ 左足を半歩後ろに下げ、右足を着地させる

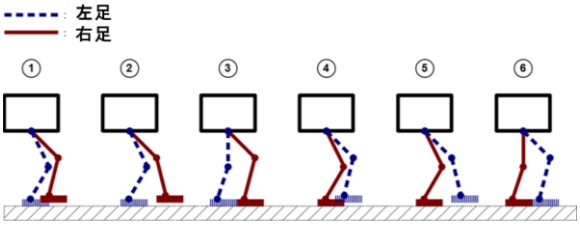

図１　歩行パターン

４．２脚歩行ロボット

前章で示した歩行を行うための脚部リンク機構を図２に示す。

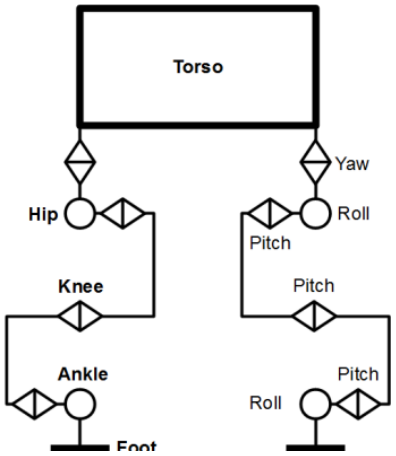

図２　脚部のリンク機構

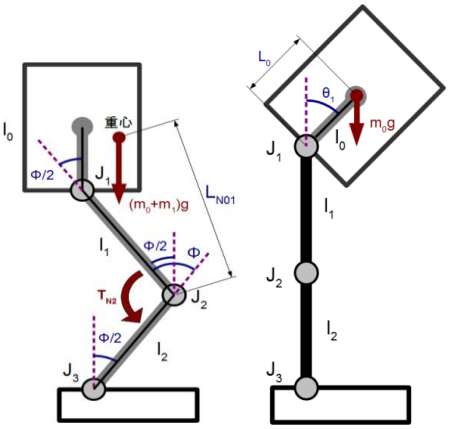

図３ 自立状態　　図４ 姿勢が崩れた状態

このロボットが図 3 の姿勢で自立した場合、膝関節と足首関節には自重によるトルクが発生する。モータの力無しでロボットが自立するには、このトルクを考慮して関節に取り付ける回転ばねの取り付け角度やばね係数を決める必要がある。

各関節にはモータと並列に回転ばねが取り付けられているとし、このロボットの自立状態で各関節にかかるトルクについて考える。このような片足のモデルにおいて、J_1 と J_3 を固定ジョイントとして剛体としたとき回転ジョイント J_2 には(1)式のトルクが発生する。

$$T_{N2} = (m_0 + m_1) g L_{N01} \sin\frac{\varphi}{2} \quad (1)$$

同様に J_1 と J_3 を固定ジョイントとした場合は次のよう

になる。

$$T_{N2} = (m_0 + m_1 + m_2)gL_{N012}\sin\frac{\varphi}{2} \quad (2)$$

ここで m_0、m_1、m_2 は各リンクの質量、L_{N01} は m_0 と m_1 を組み合わせた重心と回転関節の距離で、L_{N012} は m_0、m_1、m_2 を組み合わせた重心と回転関節との距離である。

次に図 3 から図 4 の様に直立姿勢から姿勢が崩れた場合について考える。ジョイント J_2、J_3 が固定されて剛体としているとき J_1 が θ_1[rad]だけ回転すると(3)式のトルクが発生する。

$$T_1 = m_0 g L_0 \sin\theta_1 \quad (3)$$

同様に J_1、J_3 のみを回転ジョイントとすると

$$T_1 = (m_0 + m_1)gL_{01}\sin\theta_2 \quad (4)$$

$$T_1 = (m_0 + m_1 + m_2)gL_{012}\sin\theta_3 \quad (5)$$

となる。

L_0、L_{01}、L_{012} はそれぞれの剛体の重心と回転関節の距離である。回転ばねはこれらのトルクを打ち消すように取り付けなければならない。

5．動作解析

2 脚歩行ロボットの動作を効率よく検討するために、動的モデルのシミュレーションをコンピュータ上で行う。本研究ではシミュレーションの物理計算エンジンとしてオープンソースの Open Dynamic Engine(ODE）[2]を使用した。

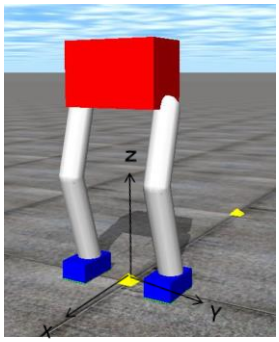

図 5　動的モデル

図 6　作成した実機

表 1　2 脚の寸法

直立姿勢時の全長[m]	0.755
直立姿勢時の脚の長さ[m]	0.555
総質量[kg]	11.65
股の幅[m]	0.200

表 2　片足の仕様

胴体リンク	長さl_0[m]	0.20
	質量m_0[kg]	6.65
腿リンク	長さl_1[m]	0.23
	質量m_1[kg]	1.20
脛リンク	長さl_2[m]	0.30
	質量m_2[kg]	1.00
足首から足先までの回転距離x_r[m]		0.09
自立姿勢時のJ_1の回転角度[rad]		π/6
自立姿勢時のJ_2の回転角度[rad]		π/3
自立姿勢時のJ_3の回転角度[rad]		π/6

表 3　自立姿勢の関節角度

股関節	Yaw[rad]	0
	Roll[rad]	0
	Pitch[rad]	π/6
膝関節	Pitch[rad]	−π/3
足首関節	Roll[rad]	0
	Pitch[rad]	π/6

表 4　各関節の仕様

		ばね 取付け角度[rad]	ばね係数 [Nm/rad]	粘性係数 [Nm s/rad]
股関節	Yaw	0.0	4.15	4.9
	Roll	0.0		
	Pitch	π/6		
膝関節	Pitch	−π/3 + 0.359	22.73	
足首関節	Pitch	π/6 − 0.776	52.67	
	Roll	0.0		

6．歩行パラメータの最適化

脚機構ロボットを歩行させるため「歩幅」、「歩行ステップ時間」、「左右重心移動量」、「足の持ち上げ高さ」などのパラメータを求めるため、2 種類の群知能アルゴリズムを用いパラメータ最適化を行った。以下に用いたアルゴリズムの概略を述べる。

6.1 ABC アルゴリズム

Artificial Bee Colony（ABC）アルゴリズム[3]は 2005 年にトルコの研究者 Dervis Karaboga により提案された。ミツバチの餌場探索を元にしたメタヒューリスティックなアルゴリズムである。ミツバチの巣の中には「Employed bee」、「Onlooker bee」、「Scout bee」の 3 つのグループがあると定義している。餌場はその座標が最適化問題の変数を、蜜の量が解の適合度を表している。

ABC アルゴリズムによる最適化の手順は以下の①～④で行う。そして指定回数、図 7 のフローのように処理する。

①パラメータ初期化および、ランダムに探索点を決定
②Employed bee が他の蜂が持つ餌場情報を元に周辺に、より良い餌場がないか探索
③Onlooker bee がルーレット選択し Employed bee により良い餌場を優先して探索
④周辺に値を更新する良い餌場が見つからなくなったら現在の餌場周辺の探索を止め、Scout bee がランダムに新しい餌場を探して記憶

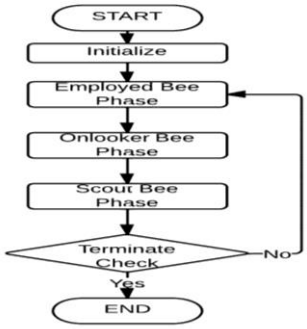

図 7　ABC アルゴリズムのフロー

6.2 PSO アルゴリズム

Particle Swarm Optimization（PSO）アルゴリズムは自然界の昆虫や鳥・魚の振る舞いを模倣している。鳥や魚は、仲間が良い餌場を見つけると次第にそこに集まり、周辺の探索と同時に探索した中での良い場所も覚えており、その周辺に戻る動きも見せる。PSO アルゴリズムはこれらの各個体を粒子で置き換え表現したものである。

本アルゴリズムの手順は図 8 のフローのように以下の①～④の処理を指定回数行う。

図 8　PSO アルゴリズムのフロー

①探索する関数に粒子をランダムに配置
②目的関数によって集団と各個体の評価値を求める
③各粒子の速度を求め、関数の新たな位置に再配置
④(②,③)をくり返し最良の評価値を求めるべく探索

7．歩行パターンの最適化

6 章で説明した 2 つの群知能アルゴリズムを用いて 15 秒間歩行させた場合の最適なパラメータを求める。表 5 に ABC アルゴリズムのパラメータを表 6 に PSO アルゴリズムのパラメータを示す。

表 5　ABC アルゴリズムパラメータ

Employed beeの数		5
Onlooker beeの数		5
最大繰り返し回数		500
許容される最大の連続探索失敗回数		100
歩行パラメータの探索範囲	歩幅[m]	[0.18,0.03]
	各歩行ステップに掛ける時間[s]	[0.10,0.05]
	左右の重心移動量[m]	[0.10,0.01]
	足を持ち上げる高さ[m]	[0.09,0.01]

表 6　PSO アルゴリズムパラメータ

粒子数		20
歩行パラメータの探索範囲	歩幅[m]	[0.03,0.18]
	各歩行ステップに掛ける時間[s]	[0.10,0.05]
	左右の重心移動量[m]	[0.10,0.01]
	足を持ち上げる高さ[m]	[0.09,0.01]

評価関数には(6)式を用いる。

$$\mathrm{fit}_{\mathrm{evl}}(x,y,z)=\frac{1}{2}\{\mathrm{sign}(z-h)+1\}\sqrt{x^2+y^2} \quad (6)$$

x,y,zは15秒後のロボットの座標を表し、$\mathrm{fit}_{\mathrm{evl}}$ (x,y,z)はその評価値を表している。(6)式右辺第一項はロボットの転倒の有無を評価する式でボディー部分の重心位置が許容値以下になった場合は転倒したと判断し全体の評価を極端に下げる様にしている。これにより、歩行可能なパラメータを戦略的に求める事が可能である。

ロバスト性を考慮して本アルゴリズムを複数回行った。そのうちの2例を表7、表8に示す。

表7　ABCアルゴリズムによる最適化の動作結果

	歩幅[m]	周期[s]	左右の重心移動量[m]	足を持ち上げ高さ[m]	移動距離[m]
Case 1	0.0497	0.0725	0.0354	0.0506	0.6197
Case 2	0.0654	0.0850	0.0387	0.0496	0.6951

表8　PSOアルゴリズムによる最適化の動作結果

	歩幅[m]	周期[s]	左右の重心移動量[m]	足を持ち上げ高さ[m]	移動距離[m]
Case 1	0.1200	0.0724	-0.0224	0.0663	0.8884
Case 2	0.0787	0.0695	-0.0336	0.0398	0.6203

これらの求められた値からの軌跡の例と転倒関数なしによる結果の移動軌跡を以下に示す。

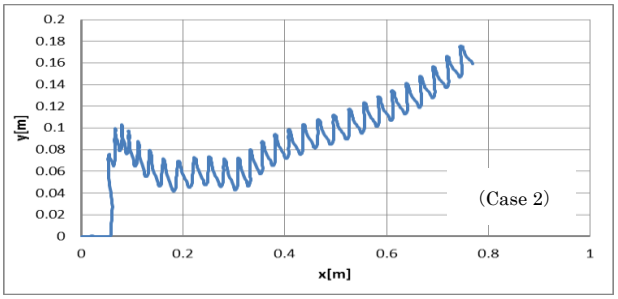

図9　ABCアルゴリズムによるxy平面上の移動軌跡

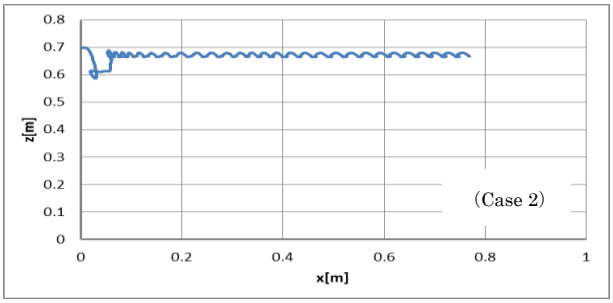

図10　ABCアルゴリズムによるxz平面上の移動軌跡

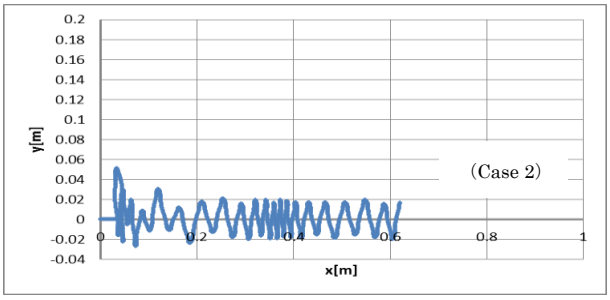

図11　PSOアルゴリズムによるxy平面上の移動軌跡

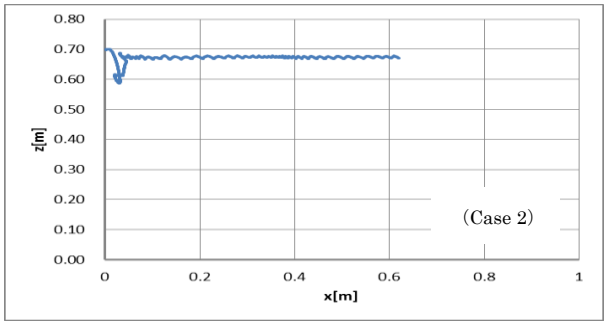

図12　PSOアルゴリズムによるxy平面上の移動軌跡

それぞれの群知能アルゴリズムで最適化を行ったパラメータで歩行シミュレーションを行った移動軌跡を図9～12に、転倒関数を用いず行った結果を図13,14に示す。

2つのアルゴリズムを比較するとABCのほうがPSOに比べ移動距離の長い適応度の高い解を導いている。転倒関数がない場合と比較すると、転倒することがなく歩行を行うデータを得ることができている。今後は評価関数に最低移動距離を設定し、それ以下の場合は評価が低くなる式にする必要があると考えられる。

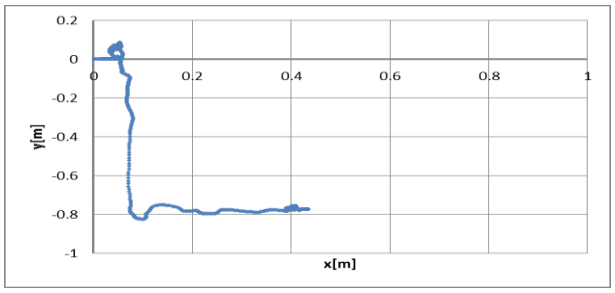

図13　転倒関数なしのxy平面上での移動軌跡

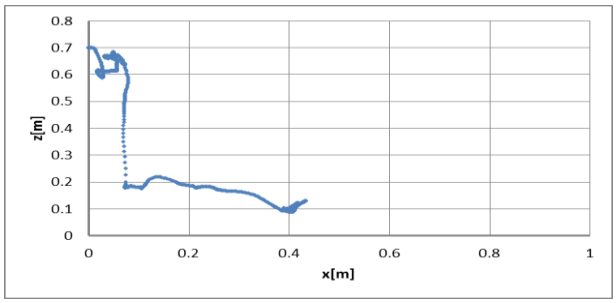

図14　転倒関数なしのxy平面上移動軌跡

８．まとめ

最適化において評価関数を適切に定めることは重要な課題であるが、複数の目的関数と重み係数の積和演算により統合しただけでは望ましい動作が得られない事が判明した。このことを考慮し、許容値を取り入れた評価関数を導入することにより転倒することなく、2脚歩行ロボットの動作制御（戦略的制御）を行うことに成功した。

[参考文献]

[1] Kiyotaka KAWAI, Kohta SUZUKI, Yuu KIMURA, and Toshihiro IRIE : Research on 2-Leg Walking Robot -Study of walking control method using optimization-, Joint 5th International Conference on Soft Computing and Intelligent Systems and 11th International Symposium on Advanced Intelligent Systems (SCIS & ISIS2010) pp.813-818,2010

[2] 出村公成: 簡単!実践!ロボットシミュレーション -Open Dynamics Engine によるロボットプログラミング, 森北出版

[3] D.Karaboga: An Idea Based on Honey Bee Swarm for Numerical Optimization, TECHNICAL REPORT-TR06, Erciyes University, Engineering Faculty, Computer Engineering.2005(PDF)

＊1　日本大学大学院理工学研究科精密機械工学専攻 大学院生
＊2　日本大学理工学部精密機械工学科　教授・博士（工学）
＊3　日本大学名誉教授・工学博士