Software Requirements Specification for Projectile

Samuel J. Crawford, Brooks MacLachlan, and W. Spencer Smith

January 9, 2024

Contents

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

The unit system used throughout is SI (Système International d'Unités). In addition to the basic units, several derived units are also used. For each unit, the Table of Units lists the symbol, a description, and the SI name.

Table 1: Table of Units

Symbol	Description	SI Name
m	length	metre
rad	angle	radian
s	time	second

1.2 Table of Symbols

The symbols used in this document are summarized in the Table of Symbols along with their units. Throughout the document, symbols in bold will represent vectors, and scalars otherwise. The symbols are listed in alphabetical order. For vector quantities, the units shown are for each component of the vector. [To avoid confusion and to allow for reuse of symbols with slightly different types (like 3D and 2D vectors), a separate table of symbols can be generated for each set of theories: background theories, helper theories, generic theories, projectile theories and final theories. —SS]

Table 2: Table of Symbols

Symbol	Description	Units
a	Scalar acceleration	$\frac{\text{m}}{\text{s}^2}$
a^c	Constant acceleration	$\frac{\text{m}}{\text{s}^2}$
a_{x}	x-component of acceleration	$\frac{\text{m}}{\text{s}^2}$
${a_{\text{x}}}^{\text{c}}$	x-component of constant acceleration	$\frac{\text{m}}{\text{s}^2}$
a_{y}	y-component of acceleration	$\frac{\text{m}}{\text{s}^2}$
${a_{\text{y}}}^{\text{c}}$	y-component of constant acceleration	$\frac{\text{m}}{\text{s}^2}$
$\mathbf{a}(t)$	Acceleration	$\frac{\text{m}}{\text{s}^2}$
$\mathbf{a}^{\text{c}}$	Constant acceleration vector	$\frac{\text{m}}{\text{s}^2}$
d_{offset}	Distance between the target position and the landing position	m

Continued on next page

Table 2: Table of Symbols (Continued)

Symbol	**Description**	**Units**
g	Magnitude of gravitational acceleration	$\frac{\mathrm{m}}{\mathrm{s}^2}$
p	Scalar position	m
$p(t)$	1D position	m
p^{i}	Initial position	m
p_{land}	Landing position	m
p_{target}	Target position	m
p_{x}	x-component of position	m
${p_{\mathrm{x}}}^{\mathrm{i}}$	x-component of initial position	m
p_{y}	y-component of position	m
${p_{\mathrm{y}}}^{\mathrm{i}}$	y-component of initial position	m
$\mathbf{p}(t)$	Position	m
s	Output message as a string	–
t	Time	s
t_{flight}	Flight duration	s
v	Speed	$\frac{\mathrm{m}}{\mathrm{s}}$
$v(t)$	1D speed	$\frac{\mathrm{m}}{\mathrm{s}}$
v^{i}	Initial speed	$\frac{\mathrm{m}}{\mathrm{s}}$
v_{launch}	Launch speed	$\frac{\mathrm{m}}{\mathrm{s}}$
v_{x}	x-component of velocity	$\frac{\mathrm{m}}{\mathrm{s}}$
${v_{\mathrm{x}}}^{\mathrm{i}}$	x-component of initial velocity	$\frac{\mathrm{m}}{\mathrm{s}}$
v_{y}	y-component of velocity	$\frac{\mathrm{m}}{\mathrm{s}}$
${v_{\mathrm{y}}}^{\mathrm{i}}$	y-component of initial velocity	$\frac{\mathrm{m}}{\mathrm{s}}$
$\mathbf{v}(t)$	Velocity	$\frac{\mathrm{m}}{\mathrm{s}}$
$\mathbf{v}^{\mathrm{i}}$	Initial velocity	$\frac{\mathrm{m}}{\mathrm{s}}$
ε	Hit tolerance	–
θ	Launch angle	rad
π	Ratio of circumference to diameter for any circle	–

1.3 Abbreviations and Acronyms

Table 3: Abbreviations and Acronyms

Abbreviation	Full Form
1D	One-Dimensional
2D	Two-Dimensional
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
PS	Physical System Description
R	Requirement
RefBy	Referenced by
Refname	Reference Name
SRS	Software Requirements Specification
TM	Theoretical Model
Uncert.	Typical Uncertainty

2 Introduction

Projectile motion is a common problem in physics. Therefore, it is useful to have a program to solve and model these types of problems. Common examples of projectile motion include ballistics problems (missiles, bullets, etc.) and the flight of balls in various sports (baseball, golf, football, etc.). The program documented here is called Projectile.

The following section provides an overview of the Software Requirements Specification (SRS) for Projectile. This section explains the purpose of this document, the scope of the requirements, the characteristics of the intended reader, and the organization of the document.

2.1 Purpose of Document

The primary purpose of this document is to record the requirements of Projectile. Goals, assumptions, theoretical models, definitions, and other model derivation information are specified, allowing the reader to fully understand and verify the purpose and scientific basis of Projectile. With the exception of system constraints, this SRS will remain abstract, describing what problem is being solved, but not how to solve it.

This document will be used as a starting point for subsequent development phases, including writing the design specification and the software verification and validation plan. The design document will show how the requirements are to be realized, including decisions on the numerical algorithms and programming environment. The verification and validation plan will show the steps that will be used to increase confidence in the software documentation and the implementation. Although the SRS fits in a series of documents that follow the so-called waterfall model, the actual development process is not constrained in any way. Even when the waterfall model is not followed, as Parnas and Clements point out [**parnasClements1986**], the most logical way to present the documentation is still to "fake" a rational design process.

2.2 Scope of Requirements

The scope of the requirements includes the analysis of a two-dimensional (2D) projectile motion problem with constant acceleration.

[We are only interested in the position of the projectile, not its orientation SD:noOrient. —SS] out, like theories of rotation

[We assume that forces are not relevant for the model so that we only need kinematic equations SD:kinOnly. —SS]

[Relative to ballistics calculations, the scope of Projectile covers relatively short distances and small magnitude initial velocities . —SS]

2.3 Characteristics of Intended Reader

Reviewers of this documentation should have an understanding of undergraduate level 1 physics and undergraduate level 1 calculus. The users of Projectile can have a lower level of expertise, as explained in Sec:User Characteristics.

2.4 Organization of Document

The organization of this document follows the template for an SRS for scientific computing software proposed by [**koothoor2013**], [**smithLai2005**], [**smithEtAl2007**], and [**smithKoothoor2016**]. The presentation follows the standard pattern of presenting goals, theories, definitions, and assumptions. For readers that would like a more bottom up approach, they can start reading the instance models and trace back to find any additional information they require.

The goal statements are refined to the theoretical models and the theoretical models to the instance models.

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics, and lists the system constraints.

Figure 1: System Context

3.1 System Context

Fig:sysCtxDiag shows the system context. A circle represents an entity external to the software, the user in this case. A rectangle represents the software system itself (Projectile). Arrows are used to show the data flow between the system and its environment.

The interaction between the product and the user is through an application programming interface. The responsibilities of the user and the system are as follows:

- User Responsibilities
  - Provide initial conditions of the physical state of the motion and the input data related to the Projectile, ensuring no errors in the data entry.
  - Ensure that consistent units are used for input variables.
  - Ensure required software assumptions are appropriate for any particular problem input to the software.
- Projectile Responsibilities
  - Detect data type mismatch, such as a string of characters input instead of a floating point number.
  - Determine if the inputs satisfy the required physical and software constraints.
  - Calculate the required outputs.

[Projectile will be used to explore different scenarios for educational and learning purposes. It will not be used to perform engineering calculations, mission-critical or safety-critical applications. —SS] [This additional context information is needed to determine how much effort should be devoted to the rationale section. If the application is safety-critical, the bar is higher. This is currently less structured, but analogous to, the idea to the Automotive Safety Integrity Levels (ASILs) that McSCert uses in their automotive hazard analyses. —SS]

3.2 User Characteristics

The end user of Projectile should have an understanding of high school physics and high school calculus.

3.3 System Constraints

There are no system constraints.

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, and definitions that are used.

4.1 Problem Description

A system is needed to predict whether a launched projectile hits its target.

4.1.1 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements.

- Launcher: Where the projectile is launched from and the device that does the launching.
- Projectile: The object to be launched at the target.
- Target: Where the projectile should be launched to.
- Gravity: The force that attracts one physical body with mass to another.
- Cartesian coordinate system: A coordinate system that specifies each point uniquely in a plane by a set of numerical coordinates, which are the signed distances to the point from two fixed perpendicular oriented lines, measured in the same unit of length (from [2]).
- Rectilinear: Occurring [fixed typo in “Ocurring” —SS] in one dimension.

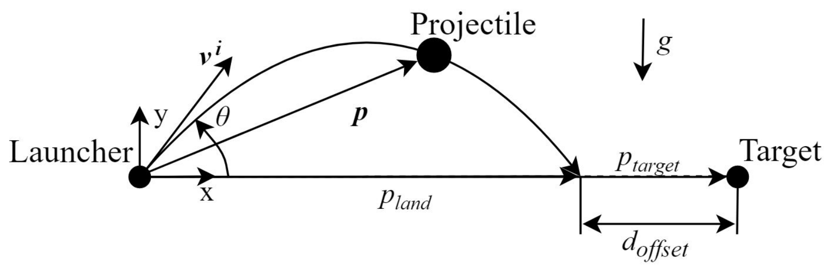

Figure 2: The physical system

4.1.2 Physical System Description

The physical system of Projectile, as shown in Fig:Launch, includes the following elements:

PS1: The launcher.

PS2: The projectile (with initial velocity $\mathbf{v}^{\text{i}}$ and launch angle θ).

PS3: The target.

4.1.3 Goal Statements

Given the initial velocity vector of the projectile and the geometric layout of the launcher and target, the goal statement is:

targetHit: Determine if the projectile hits the target.

4.2 Solution Characteristics Specification

The instance models that govern Projectile are presented in the Instance Model Section. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.2.1 [Types —SS]

[time = $\mathbb{R}$ —SS]

4.2.2 [Scope Decisions —SS]

SD:noOrient: [The orientation of the projectile is ignored. We only care about its translation not its rotation. (RefBy: Scope of Requirements). —SS]

SD:kinOnly: [The motion of the projectile is modelled with only kinematic equations. Forces are not considered. (RefBy: Scope of Requirements). —SS]

4.2.3 [Modelling Decisions —SS]

[The modelling decisions should likely eventually use a prefix like MD, but for now most of them will use the prefix A, since that is what was previously used. —SS]

MD:cartSyst: A Cartesian coordinate system is used. (RefBy: TM:acceleration, TM:velocity, TM:directionCosines, GD:rectVel, GD:rectPos, GD:velVec, GD:posVec, GD:magAngleToCompRep, PT:coordSyst, DD:speedIX, DD:speedIY, PT:velVecInitMagAndAngle, PT:posVecInitMagAndAngle, PT:posVecInitPos, PT:velVecPlanetaryGrav, PT:posVecPlanetaryGrav, IM:calOfLandingTime, IM:calOfLandingDist, IM:offsetIM, and IM:messageIM.)

yAxisGravity: The direction of the y-axis is directed opposite to gravity. (RefBy: IM:calOfLandingDist, IM:calOfLandingTime, IM:messageIM, PT:posVecInitPos, IM:offsetIM, PT:posVecPlanetaryGrav, PT:velVecPlanetaryGrav, PT:coordSyst, PT:posVecInitMagAndAngle, and PT:velVecInitMagAndAngle.)

launchOrigin: The launcher is coincident with the origin. (RefBy: IM:calOfLandingDist and IM:calOfLandingTime.)

targetXAxis: The target lies on the x-axis (from A:neglectCurv). (RefBy: IM:calOfLandingTime.)

posXDirection: The positive x-direction is from the launcher to the target. (RefBy: IM:offsetIM, IM:messageIM, IM:calOfLandingDist, and IM:calOfLandingTime.)

timeStartZero: Time starts at zero. (RefBy: GD:velVec, GD:rectVel, GD:rectPos, GD:posVec, and IM:calOfLandingTime.)

D:towardLauncher: [The launch velocity is positive. (RefBy: PT:velVecInitMagAndAngle) —SS]

MD:magAngleRep: [Use the magnitude and angle representation for the initial velocity vector. (RefBy: PT:velVecInitMagAndAngle) —SS]

4.2.4 [Background Theory Assumptions —SS]

threeD: [The motion of the body is in all three dimensions. —SS] (RefBy: TM:acceleration, TM:velocity, TM:directionCosines)

4.2.5 [Helper Theory Assumptions (GD:rectVel, GD:rectPos) —SS]

oneD: The motion of the particle is one dimensional. (RefBy: GD:velVec and GD:posVec.)

constAccel: The acceleration is constant (from A:accelXZero, A:accelYGravity, A:neglectDrag, and A:noObstruct). (RefBy: GD:velVec and GD:posVec.)

4.2.6 [Generic Theory Assumptions (GD:velVec, GD:posVec) —SS]

twoD: The variables only depend on two-dimensions (2D). (RefBy: GD:velVec and GD:posVec.) [changed twoDMotion to just twoD, so that it can be used for things that are 2D other than just the motion. —SS]

constAccelX: [The acceleration is constant in the x direction. (RefBy: GD:velVec and GD:posVec.) —SS]

constAccelY: [The acceleration is constant in the y direction. (RefBy: GD:velVec and GD:posVec.) —SS]

gravAccel: [The acceleration is due to gravity. (RefBy: PT:velVecPlanetaryGrav and PT:posVecPlanetaryGrav.) —SS]

4.2.7 [Projectile Specific Theory Assumptions —SS]

accelXZero: The constant acceleration in the x-direction is zero. (RefBy: IM:calOfLandingDist and A:constAccel.)

accelYGravity: The constant acceleration in the y-direction is the acceleration due to gravity (from A:yAxisGravity). (RefBy: IM:calOfLandingTime and A:constAccel).

gravAccelValue: The acceleration due to gravity is assumed to have the value provided in the section for Values of Auxiliary Constants. (RefBy: IM:calOfLandingDist and IM:calOfLandingTime.)

noObstruct: No obstructions impede the path of the projectile. This means there is nothing rising from the ground that will block the flight and no other projectiles collide with the launched projectile. (RefBy: PT:coordSyst, IM:calOfLandingDist, etc.)

flatPlanet: No surface of the planet is assummed flat; that is, there is assumed to be no curvature. (RefBy: PT:coordSyst, IM:calOfLandingDist, etc.)

4.2.8 [Rationale Assumptions —SS]

centreMass: We only care about the motion of the centre of mass of the projectile. (RefBy: Section 5.3.1)

neglectCurv: The distance is small enough that the curvature of the celestial body can be neglected. (RefBy: A:targetXAxis and MD:cartSyst.)

neglectDrag: Air drag is neglected. (RefBy: A:constAccel.)

negPlanetRot: [The rotation of the planet is neglected —SS]. (RefBy: ?)

4.2.9 [Context Theories —SS]

[In science and engineering the theories that are used to solve practical problems are built on a foundation of mathematical and physical knowledge. For practical reasons, it is not usually feasible to define all of the details down to the most fundamental level. For instance, the audience reading the requirements will typically know the concepts of real arithmetic, differentiation, trigonometry, etc. Nevertheless it is helpful to make the dependence on fundamental theories explicit, even if those theories themselves are not explicitly defined. These theories form the context in which the higher level theories exist. The context theories that are used for the Projectile problem are listed below. —SS]

Refname	**[CT:realArith —SS]**
Label	Real Arithmetic
Source	[**<empty citation>**]
RefBy	TM:acceleration, TM:velocity, TM:directionCosines, GD:rectVel, GD:rectVel, DD:speedIX, DD:speedIY,everything (TODO: not complete; almost every theory uses this context theory.)

Refname	**[CT:trigonometry —SS]**
Label	Trigonometry
Source	[**<empty citation>**]
RefBy	TM:directionCosines, DD:speedIX, DD:speedIY,everything (TODO: fill in)

Refname	**[CT:vectors —SS]**
Label	Vectors
Source	[**<empty citation>**]
RefBy	TM:acceleration, TM:velocity, TM:directionCosines, GD:rectVel, GD:rectVel, DD:speedIX, DD:speedIY,TODO: fill in

Refname	**[CT:CartCoordSyst —SS]**
Label	Cartesian Coordinate System
Source	[**<empty citation>**]
RefBy	TM:acceleration, TM:velocity, TM:directionCosines, GD:rectVel, GD:rectVel, DD:speedIX, DD:speedIY,TODO: fill in

Refname	**[CT:Differentiation —SS]**
Label	Differentiation
Source	[**<empty citation>**]
RefBy	TM:acceleration, TM:velocity, GD:rectVel, GD:rectVel, TODO: fill in

Refname	**[CT:Integration —SS]**
Label	Integration
Source	[**<empty citation>**]
RefBy	GD:rectVel, TODO: fill in

4.2.10 [Background Theories (BT) —SS]

This section focuses on the general equations and laws that Projectile is based on. [Maybe relabel all of the background theories with the prefix BT, instead of TM? —SS]

Refname	**TM:acceleration**
Label	Acceleration
Equation	$$\mathbf{a}(t) = \frac{d\mathbf{v}(t)}{dt}$$
Description	[t : time —SS] is the time (s) [$\mathbf{a}$: time $\rightarrow \mathbb{R}^3$ —SS] is the acceleration ($\frac{\text{m}}{\text{s}^2}$) [$\mathbf{v}$: time $\rightarrow \mathbb{R}^3$ —SS] is the velocity ($\frac{\text{m}}{\text{s}}$)
[Constraints —SS]	[None —SS]
Notes	The velocity and acceleration of the body are expressed using a 3D (A:threeD) Cartesian coordinate system (CT:CartCoordSyst, MD:cartSyst). That is, the coordinate system is rectangular (orthonormal). The relationship between acceleration and velocity uses the concepts of real arithmetic, vectors and differentiation (CT:realArith, CT:vectors, CT:Differentiation).
Source	[1]
RefBy	GD:rectVel

Context Theories Used by TM:acceleration

- CT:realArith
- CT:vectors
- CT:CartCoodSyst
- CT:Differentiation

Initial Theories Used by TM:acceleration

None

Preconditions for TM:acceleration

- MD:cartSyst

- A:threeD

Refname	**TM:velocity**
Label	Velocity
Equation	$$\mathbf{v}(t) = \frac{d\mathbf{p}(t)}{dt}$$
Description	[t : time —SS] is the time (s) [$\mathbf{v}$: time $\rightarrow \mathbb{R}^3$ —SS] is the velocity ($\frac{\mathrm{m}}{\mathrm{s}}$) [$\mathbf{p}$: time $\rightarrow \mathbb{R}^3$ —SS] is the position (m)
[Constraints —SS]	[None —SS]
Notes	The position and velocity of the body are expressed using a 3D (A:threeD) Cartesian coordinate system (CT:CartCoordSyst, MD:cartSyst). That is, the coordinate system is rectangular (orthonormal). The relationship between velocity and position uses the concepts of real arithmetic, vectors and differentiation (CT:realArith, CT:vectors, CT:Differentiation).
Source	[3]
RefBy	GD:rectPos

Context Theories Used by TM:velocity

- CT:realArith
- CT:vectors
- CT:CartCoodSyst
- CT:Differentiation

Initial Theories Used by TM:velocity

None

Preconditions for TM:velocity

- MD:cartSyst
- A:threeD

Refname	TM:directionCosines
Label	[Direction Cosines Representation for Vectors —SS]
Equation	$$b_x = \|\mathbf{b}\| \cos(\alpha), b_y = \|\mathbf{b}\| \cos(\beta), b_z = \|\mathbf{b}\| \cos(\gamma)$$
Description	[$\alpha : \mathbb{R}$ —SS] is the angle between the vector and the positive x axis [$\beta : \mathbb{R}$ —SS] is the angle between the vector and the positive y axis [$\gamma : \mathbb{R}$ —SS] is the angle between the vector and the positive z axis [$\|\mathbf{b}\| : \mathbb{R}$ —SS] is the magnitude of the vector [$b_x : \mathbb{R}$ —SS] is the x component of the vector $\mathbf{b}$ [$b_y : \mathbb{R}$ —SS] is the y component of the vector $\mathbf{b}$ [$b_z : \mathbb{R}$ —SS] is the z component of the vector $\mathbf{b}$
[Constraints —SS]	[$\cos^2(\alpha) + \cos^2(\beta) + \cos^2(\gamma) = 1$ —SS]
Notes	[The vector **b** is in a 3D (A:threeD) Cartesian coordinate system (CT:CartCoordSyst, MD:cartSyst). —SS] [A figure showing the angles for a sample vector would be a nice addition. —SS] Direction cosines use the context theories of real arithmetic, trigonometry and vectors (CT:realArith, CT:trigonometry, CT:vectors).
Source	[**<empty citation>**] web-page resource, Long book
RefBy	GD:magAngleToCompRep

Context Theories Used by TM:directionCosines

- CT:realArith
- CT:trigonometry
- CT:CartCoordSyst
- CT:vectors

Initial Theories Used by TM:directionCosines

None

Preconditions for TM:directionCosines

- MD:cartSyst
- A:threeD

4.2.11 [Helper Theories (GD:rectVel and GD:rectPos) —SS]

This section collects the laws and equations that will be used to build the instance models. [We should remove the prefix GD. Maybe we should replace it with the prefix TM? —SS]

Refname	GD:rectVel
Label	Rectilinear (1D) velocity as a function of time for constant acceleration
Units	$\frac{\text{m}}{\text{s}}$ [Do we need units as a separate field? user option? —SS]
Equation	$$v(t) = v^{\text{i}} + a^{c} t$$
Description	[v : time $\rightarrow \mathbb{R}$ —SS] is the 1D speed ($\frac{\text{m}}{\text{s}}$) [$v^{\text{i}}$: $\mathbb{R}$ —SS] is the initial speed ($\frac{\text{m}}{\text{s}}$) [$a^{c}$: $\mathbb{R}$ —SS] is the constant acceleration ($\frac{\text{m}}{\text{s}^2}$) [$t$: time —SS] is the time (s)
[Constraints —SS]	[$t \geq 0$ —SS] [A:timeStartZero —SS]
[Notes —SS]	[See detailed derivation of equation and constraint below —SS]
Source	[5, (pg. 8)]
RefBy	GD:velVec

Detailed derivation of rectilinear velocity: [We start from the theory of acceleration TM:acceleration for a body in 3D Cartesian space: —SS]

$$\mathbf{a}(t) = \frac{d\mathbf{v}(t)}{dt}$$

[At this point we change the assumption of 3D motion (A:threeD) to instead assume the body only travels in a straight line along one dimension of the coordinate system (A:oneD): —SS]

$$a(t) = \frac{dv(t)}{dt}$$

[We now assume that the acceleration is constant (does not vary with time) (A:constAccel) represented by a^c. The initial velocity (at $t = 0$, from A:timeStartZero) is represented by v^{i}. We now have: —SS]

$$a^c = \frac{dv}{dt}$$

Rearranging and integrating, we have:

$$\int_{v^{\mathrm{i}}}^{v} 1 \, dv = \int_{0}^{t} a^c \, dt$$

Performing the integration, we have the required equation:

$$v(t) = v^{\mathrm{i}} + a^c t$$

[The theory for rectilinear velocity uses the context theories from TM:acceleration of real arithmetic, vectors, Cartesian coodinate system, and differentiation (CT:realArith, CT:vectors, CT:CartCoordSyst, CT:Differentiation). In addition, the context theory of integration is used (CT:Integration). —SS]

Detailed derivation of constraints for GD:rectVel: [The data constraint that $t \geq 0$ comes from the assumption that times starts at zero (A:timeStartZero); that is, the initial velocity applies at $t = 0$. Therefore, the equation is nonsensical for negative time. —SS]

Context Theories Used by GD:rectVel

- CT:realArith
- CT:vectors
- CT:CartCoordSyst
- CT:Differentiation
- CT:Integration

Initial Theories Used by GD:rectVel

- TM:acceleration

[Preconditions for GD:rectVel —SS]

- MD:cartSyst (inherited from TM:acceleration)
- A:oneD (overrides A:threeD)
- A:constAccel
- A:timeStartZero

Refname	**GD:rectPos**
Label	Rectilinear (1D) position as a function of time for constant acceleration
Units	m
Equation	$$p(t) = p^{\text{i}} + v^{\text{i}}t + \frac{a^c t^2}{2}$$
Description	[p : time $\rightarrow \mathbb{R}$ —SS] is the 1D position (m) [$p^{\text{i}} : \mathbb{R}$ —SS] is the initial position (m) [$v^{\text{i}} : \mathbb{R}$ —SS] is the initial speed ($\frac{\text{m}}{\text{s}}$) [$t : \mathbb{R}$ —SS] is the time (s) [$a^c : \mathbb{R}$ —SS] is the constant acceleration ($\frac{\text{m}}{\text{s}^2}$)
[Constraints —SS]	[$t \geq 0$ —SS] [A:timeStartZero —SS]
[Notes —SS]	[See detailed derivation of equation and constraint below —SS]
Source	[5, (pg. 8)]
RefBy	GD:posVec

Detailed derivation of rectilinear position: [We start from the kinematic equation for velocity TM:velocity for a body in 3D Cartesian space: —SS]

$$\mathbf{v}(t) = \frac{d\mathbf{p}(t)}{dt}$$

[At this point we assume the body only travels in a straight line along one dimension of the coordinate system (A:oneD): —SS]

$$v(t) = \frac{dp(t)}{dt}$$

[The initial position (at $t = 0$, from A:timeStartZero) is represented by p^{i}. Rearranging the above equation and integrating we have: —SS]

$$\int_{p^{\mathrm{i}}}^{p(t)} 1\,dp = \int_0^t v(t)\,dt$$

[This equation has been changed from the previous version to show $p(t)$ and $v(t)$ —SS]

[We now assume that the acceleration is constant (does not vary with time) (A:constAccel) represented by a^c. The initial velocity (at $t = 0$, from A:timeStartZero) is represented by v^{i}. Since we satisfy the preconditions for GD:rectVel we can replace $v(t)$ to get: —SS]

$$\int_{p^{\mathrm{i}}}^{p(t)} 1\,dp = \int_0^t v^{\mathrm{i}} + a^c t\,dt$$

[The above equation has been changed from the previous version to show $p(t)$. —SS] Performing the integration, we have the required equation:

$$p(t) = p^{\mathrm{i}} + v^{\mathrm{i}}t + \frac{a^c t^2}{2}$$

[The theory for rectilinear position uses the context theories from TM:acceleration of real arithmetic, vectors, Cartesian coodinate system, and differentiation (CT:realArith, CT:vectors, CT:CartCoordSyst, CT:Differentiation). In addition, the context theory of integration is used (CT:Integration). —SS]

Detailed derivation of constraints for GD:rectPos: [The data constraint that $t \geq 0$ comes from the assumption that times starts at zero (A:timeStartZero); that is, the initial velocity applies at $t = 0$. Therefore, the equation is nonsensical for negative time. —SS]

Context Theories Used by GD:rectPos

- CT:realArith
- CT:vectors
- CT:CartCoordSyst
- CT:Differentiation

- CT:Integration

Initial Theories Used by GD:rectPos

- TM:velocity

Preconditions for GD:rectPos

- MD:cartSyst (inherited from TM:velocity)
- A:oneD
- A:constAccel
- A:timeStartZero

4.2.12 [Generic Theories (GD:velVec and GD:posVec) —SS]

Refname	**GD:velVec**
Label	Velocity vector as a function of time for 2D motion under constant acceleration
Units	$\frac{\text{m}}{\text{s}}$
Equation	$$\mathbf{v}(t) = \begin{bmatrix} {v_{\text{x}}}^{\text{i}} + {a_{\text{x}}}^{\text{c}}t \\ {v_{\text{y}}}^{\text{i}} + {a_{\text{y}}}^{\text{c}}t \end{bmatrix}$$
Description	[$\mathbf{v}$: time $\rightarrow \mathbb{R}^2$ —SS] is the velocity [vector —SS] ($\frac{\text{m}}{\text{s}}$) [${v_{\text{x}}}^{\text{i}} : \mathbb{R}$ —SS] is the x-component of initial velocity ($\frac{\text{m}}{\text{s}}$) [${a_{\text{x}}}^{\text{c}} : \mathbb{R}$ —SS] is the x-component of constant acceleration ($\frac{\text{m}}{\text{s}^2}$) [$t$: time —SS] is the time (s) [${v_{\text{y}}}^{\text{i}} : \mathbb{R}$ —SS] is the y-component of initial velocity ($\frac{\text{m}}{\text{s}}$) [${a_{\text{y}}}^{\text{c}} : \mathbb{R}$ —SS] is the y-component of constant acceleration ($\frac{\text{m}}{\text{s}^2}$)
[Constraints —SS]	[$t \geq 0$ —SS] [Inherited from GD:rectVel —SS]
Source	–
RefBy	PT:velVecInitMagAndAngle

Detailed derivation of velocity vector: For a two-dimensional Cartesian coordinate system (A:twoD and MD:cartSyst), we can represent the velocity vector as $\mathbf{v}(t) = \begin{bmatrix} v_{\text{x}}([t - - - SS]) \\ v_{\text{y}}([t - - - SS]) \end{bmatrix}$ and the acceleration vector as $\mathbf{a}(t) = \begin{bmatrix} a_{\text{x}}([t - - - SS]) \\ a_{\text{y}}([t - - - SS]) \end{bmatrix}$. The acceleration is assumed to be constant in both the x direction ${a_{\text{x}}}^{\text{c}}$ (A:constAccelX) and y direction ${a_{\text{y}}}^{\text{c}}$ (A:constAccelY). The constant acceleration vector is represented as $\mathbf{a}^{\text{c}} = \begin{bmatrix} {a_{\text{x}}}^{\text{c}} \\ {a_{\text{y}}}^{\text{c}} \end{bmatrix}$. The initial velocity (at $t = 0$, from A:timeStartZero) is represented by $\mathbf{v}^{\text{i}} = \begin{bmatrix} {v_{\text{x}}}^{\text{i}} \\ {v_{\text{y}}}^{\text{i}} \end{bmatrix}$. [For each coordinate direction we satisfy the preconditions for GD:rectVel (MD:cartSyst, A:oneD, A:constAccel, A:timeStartZero). This means we can use the one dimensional equation for each of the two coordinate directions to yield the required equation: —SS]

$$\mathbf{v}(t) = \begin{bmatrix} {v_{\text{x}}}^{\text{i}} + {a_{\text{x}}}^{\text{c}} t \\ {v_{\text{y}}}^{\text{i}} + {a_{\text{y}}}^{\text{c}} t \end{bmatrix}$$

[The theory for the velocity vector for rectilinear motion in 2D uses the context theories from GD:rectVel of real arithmetic, vectors, Cartesian coodinate system, differentiation and integration (CT:realArith, CT:vectors, CT:CartCoordSyst, CT:Differentiation, CT:Integration). —SS]

Context Theories Used by GD:velVec

- CT:realArith
- CT:vectors
- CT:CartCoordSyst
- CT:Differentiation
- CT:Integration

Initial Theories Used by GD:velVec

- GD:rectVel

[If a theory A depends another theory B and theory B uses context theory C, does theory A also depend on context theory C, even if context theory C never appears in the expression or derivation of theory A? —SS]

Preconditions for GD:velVec

- A:timeStartZero (inherited from GD:rectVel)
- MD:cartSyst (inherited from GD:rectVel)

- A:twoD (A:oneD in both the x and y directions)
- A:constAccelX (A:constAccel in x direction)
- A:constAccelY (A:constAccel in y direction)

[The assumptions (preconditions) for GD:velVec are not a superset of the assumptions for the theories it is based on. Specifically, A:constAccel becomes two assumptions: A:constAccel and A:constAccelY. Moreover, the assumption A:oneD becomes A:twoD. —SS]

Refname	**GD:posVec**
Label	Position vector as a function of time for 2D motion under constant acceleration
Units	m
Equation	$$\mathbf{p}(t) = \begin{bmatrix} {p_{\text{x}}}^{\text{i}} + {v_{\text{x}}}^{\text{i}} t + \frac{{a_{\text{x}}}^{\text{c}} t^2}{2} \\ {p_{\text{y}}}^{\text{i}} + {v_{\text{y}}}^{\text{i}} t + \frac{{a_{\text{y}}}^{\text{c}} t^2}{2} \end{bmatrix}$$
Description	[$\mathbf{p}$: time $\rightarrow \mathbb{R}$ —SS] is the position [vector —SS] (m) [${p_{\text{x}}}^{\text{i}} : \mathbb{R}$ —SS] is the x-component of initial position (m) [${v_{\text{x}}}^{\text{i}} : \mathbb{R}$ —SS] is the x-component of initial velocity ($\frac{\text{m}}{\text{s}}$) [$t$: time —SS] is the time (s) [${a_{\text{x}}}^{\text{c}} : \mathbb{R}$ —SS] is the x-component of constant acceleration ($\frac{\text{m}}{\text{s}^2}$) [${p_{\text{y}}}^{\text{i}} : \mathbb{R}$ —SS] is the y-component of initial position (m) [${v_{\text{y}}}^{\text{i}} : \mathbb{R}$ —SS] is the y-component of initial velocity ($\frac{\text{m}}{\text{s}}$) [${a_{\text{y}}}^{\text{c}} : \mathbb{R}$ —SS] is the y-component of constant acceleration ($\frac{\text{m}}{\text{s}^2}$)
[Constraints —SS]	[$t \geq 0$ —SS] [Inherited from GD:rectPos —SS]
Source	–
RefBy	PT:posVecInitMagAndAngle

Detailed derivation of position vector: For a two-dimensional Cartesian coordinate system (A:twoD and MD:cartSyst), we can represent the position vector as $\mathbf{p}(t) = \begin{bmatrix} p_{\text{x}}[(t) - - - SS] \\ p_{\text{y}}[(t) - - - SS] \end{bmatrix}$, the velocity vector as $\mathbf{v}(t) = \begin{bmatrix} v_{\text{x}} \\ v_{\text{y}} \end{bmatrix}$, and the acceleration vector as $\mathbf{a}(t) = \begin{bmatrix} a_{\text{x}}[(t) - - - SS] \\ a_{\text{y}}[(t) - - - SS] \end{bmatrix}$. The acceleration is assumed to be constant (A:constAccel) and the constant acceleration vector is represented as $\mathbf{a}^{\text{c}} = \begin{bmatrix} {a_{\text{x}}}^{\text{c}} \\ {a_{\text{y}}}^{\text{c}} \end{bmatrix}$. The initial velocity (at $t = 0$, from A:timeStartZero) is represented by $\mathbf{v}^{\text{i}} = \begin{bmatrix} {v_{\text{x}}}^{\text{i}} \\ {v_{\text{y}}}^{\text{i}} \end{bmatrix}$. [For each coordinate direction we satisfy the preconditions for GD:rectPos (MD:cartSyst, A:oneD, A:constAccel, A:timeStartZero). This means we can use the one dimensional equation for each of the two coordinate directions to yield the required equation: —SS]

$$\mathbf{p}(t) = \begin{bmatrix} {p_{\text{x}}}^{\text{i}} + {v_{\text{x}}}^{\text{i}} t + \frac{{a_{\text{x}}}^{\text{c}} t^2}{2} \\ {p_{\text{y}}}^{\text{i}} + {v_{\text{y}}}^{\text{i}} t + \frac{{a_{\text{y}}}^{\text{c}} t^2}{2} \end{bmatrix}$$

[The theory for the position vector for rectilinear motion in 2D uses the context theories from GD:rectPos of real arithmetic, vectors, Cartesian coodinate system, differentiation and integration (CT:realArith, CT:vectors, CT:CartCoordSyst, CT:Differentiation, CT:Integration). —SS]

Theories Used by GD:posVec

- CT:realArith
- CT:vectors
- CT:CartCoordSyst
- CT:Differentiation
- CT:Integration

Initial Theories Used by GD:posVec

- GD:rectPos

Preconditions for GD:posVec

- A:timeStartZero (inherited from GD:rectPos)
- MD:cartSyst (inherited from GD:rectPos)
- A:twoD (A:oneD in both the x and y directions)
- A:constAccelX (A:constAccel in x direction)
- A:constAccelY (A:constAccel in y direction)

Refname	GD:magAngleToCompRep
Label	[Conversion of Magnitude and Angle Representation of a Vector to the Component Representation —SS]
Equation	$$b_x = \|\mathbf{b}\| \cos(\theta), b_y = \|\mathbf{b}\| \sin(\theta)$$
Description	[θ : $\mathbb{R}$ —SS] is the angle between the vector and the positive x axis ([θ means something more general here. Should a different symbol be used? —SS]) [$\|\mathbf{b}\|$: $\mathbb{R}$ —SS] is the magnitude of the vector [b_x : $\mathbb{R}$ —SS] is the x component of the vector $\mathbf{b}$ [b_y : $\mathbb{R}$ —SS] is the y component of the vector $\mathbf{b}$
[Constraints —SS]	[None —SS]
Notes	[The vector $\mathbf{b}$ is in a two dimensional (A:twoD) Cartesian coordinate system (MD:cartSyst). —SS] [The equations can be derived from TM:directionCosines for a 2D sytem. In a 2D system, the angle γ is not relevant because in this case $\gamma = \pi/2$, and $\cos(\gamma) = \cos(\pi/2) = 0$. For the 2D case we rename the angle α as θ. The angle β is related to θ by $\beta = \pi/2 - \theta$; therefore, $\cos(\beta) = \cos(\pi/2 - \theta) = \sin(\theta)$. —SS] This theory uses the same context theories as TM:directionCosines: CT:realArith, CT:trigonometry, CT:CartCoodSyst and CT:vectors.
Source	[**<empty citation>**]
RefBy	DD:speedIX, DD:speedIY

Context Theories Used by GD:magAngleToCompRep

- CT:realArith
- CT:trigonometry
- CT:vectors
- CT:CartCoordSyst

Initial Theories Used by GD:magAngleToCompRep

- TM:directionCosines

Preconditions for GD:magAngleToCompRep

- MD:cartSyst (inherited from directionCosines)
- A:twoD

4.2.13 [Projectile Theories (List Them) —SS]

[Remove this “This section collects and defines all the data needed to build the instance models.” —SS]

[The general theories are now refined into specific projectile theories. A specific coordinate system is introduced. —SS]

[In this section the generic “body” becomes projectile. Should we refine that in code, or just rely on the user to change the teminology? —SS]

Refname	PT:coordSyst
Label	Coordinate System for Projectile
Symbol	[none —SS]
Units	[none —SS]
Equation	[none —SS]
Description	[none —SS]
Notes	[The coordinate system is shown in Fig:Launch. As the figure shows, the origin of the 2D (A:twoD) Cartesian coordinate system (MD:cartSyst, CT:CartCoordSyst) coincides with the location of the Launcher (A:launchOrigin). The Target lies on the x-axis (A:targetXAxis) and the positive x-direction is from the launcher to the target (A:posXDirection). The positive y-direction is up A:yAxisGravity. —SS] The planet is flat (A:flatPlanet); that is, the curvature of the planet is ignored. Moreover, there are no obstructions blocking the path of the projectile (A:noObstruct).
Source	–
RefBy	DD:speedIX, DD:speedIY

Context Theories Used by PT:coordSyst

- CT:CartCoordSyst

Initial Theories Used by PT:coordSyst

None

Preconditions for PT:coordSyst

- MD:cartSyst
- A:twoD
- A:launchOrigin

- A:targetXAxis
- A:posXDirection
- A:yAxisGravity
- A:noObstruct
- A:flatPlanet

Refname	**DD:vecMag** [REMOVE —SS]
Label	Speed

Refname	**DD:speedIX**
Label	x-component of initial velocity
Symbol	${v_{\text{x}}}^{\text{i}}$
Units	$\frac{\text{m}}{\text{s}}$
Equation	$${v_{\text{x}}}^{\text{i}} = v^{\text{i}} \cos\left(\theta\right)$$
Description	[${v_{\text{x}}}^{\text{i}} : \mathbb{R}$ —SS] is the x-component of initial velocity ($\frac{\text{m}}{\text{s}}$) [$v^{\text{i}} : \mathbb{R}$ —SS] is the initial speed ($\frac{\text{m}}{\text{s}}$) [$\theta : \mathbb{R}$ —SS] is the launch angle (rad)
Notes	[This equation is a relabelling of the x component of GD:magAngleToCompRep — v^{i} is $\lvert\mathbf{b}\rvert$, ${v_{\text{x}}}^{\text{i}}$ is b_x and θ is θ. —SS] θ is shown in Fig:Launch. This equation uses the projectile coordinate system (PT:coordSyst) and inherits the assumptions from GD:magAngleToCompRep: MD:cartSyst and A:twoD.
Source	–
RefBy	[PT:posVecInitMagAndAngle, PT:velVecInitMagAndAngle —SS]

Context Theories Used by DD:speedIX

- CT:realArith
- CT:trigonometry
- CT:CartCoordSyst
- CT:vectors

Initial Theories Used by DD:speedIX

- GD:magAngleToCompRep

Preconditions for DD:speedIX

- MD:cartSyst (inherited from GD:magAngleToCompRep)
- A:twoD (inherited from GD:magAngleToCompRep)
- A:launchOrigin (inherited from PT:coordSyst)
- A:posXDirection (inherited from PT:coordSyst)
- A:targetXAxis (inherited from PT:coordSyst)
- A:yAxisGravity (inherited from PT:coordSyst)
- A:noObstruct (inherited from PT:coordSyst)
- A:flatPlanet

Refname	**DD:speedIY**
Label	y-component of initial velocity
Symbol	${v_{\text{y}}}^{\text{i}}$
Units	$\frac{\text{m}}{\text{s}}$
Equation	$${v_{\text{y}}}^{\text{i}}=v^{\text{i}}\sin\left(\theta\right)$$
Description	[${v_{\text{y}}}^{\text{i}}$: $\mathbb{R}$ —SS] is the y-component of initial velocity ($\frac{\text{m}}{\text{s}}$) [$v^{\text{i}}$: $\mathbb{R}$ —SS] is the initial speed ($\frac{\text{m}}{\text{s}}$) [$\theta$: $\mathbb{R}$ —SS]is the launch angle (rad)
Notes	[This equation is a relabelling of the x component of GD:magAngleToCompRep — v^{i} is $\|\mathbf{b}\|$, ${v_{\text{y}}}^{\text{i}}$ is b_y and θ is θ. —SS] θ is shown in Fig:Launch. This equation inherits the assumptions from GD:magAngleToCompRep: MD:cartSyst and A:twoD.
Source	–
RefBy	[PT:posVecInitMagAndAngle, PT:velVecInitMagAndAngle —SS]

Context Theories Used by DD:speedIY

- CT:realArith
- CT:vectors
- CT:CartCoordSyst

Initial Theories Used by DD:speedIY

- GD:magAngleToCompRep

Preconditions for DD:speedIY

- MD:cartSyst (inherited from GD:magAngleToCompRep)

- A:twoD (inherited from GD:magAngleToCompRep)
- A:launchOrigin (inherited from PT:coordSyst)
- A:posXDirection (inherited from PT:coordSyst)
- A:targetXAxis (inherited from PT:coordSyst)
- A:yAxisGravity (inherited from PT:coordSyst)
- A:noObstruct (inherited from PT:coordSyst) A:flatPlanet

Refname	PT:velVecInitMagAndAngle
Label	Velocity vector as a function of time for 2D projectile motion under constant acceleration in both the x and y directions using the magnitude and angle representation of the initial velocity.
Units	$\frac{\mathrm{m}}{\mathrm{s}}$
Equation	$$\mathbf{v}(t) = \begin{bmatrix} v_{\text{launch}} \cos(\theta) + {a_{\text{x}}}^{\text{c}} t \\ v_{\text{launch}} \sin(\theta) + {a_{\text{y}}}^{\text{c}} t \end{bmatrix}$$
Description	[$\mathbf{v}$: time $\rightarrow \mathbb{R}^2$ —SS] is the velocity [vector —SS] ($\frac{\mathrm{m}}{\mathrm{s}}$) [$v_{\text{launch}} : \mathbb{R}$ —SS] is the initial speed ($\frac{\mathrm{m}}{\mathrm{s}}$) [${a_{\text{x}}}^{\text{c}} : \mathbb{R}$ —SS] is the x-component of constant acceleration ($\frac{\mathrm{m}}{\mathrm{s}^2}$) [$t$: time —SS] is the time (s) [${a_{\text{y}}}^{\text{c}} : \mathbb{R}$ —SS] is the y-component of constant acceleration ($\frac{\mathrm{m}}{\mathrm{s}^2}$) $\theta : \mathbb{R}$ is the launch angle (rad)
[Constraints —SS]	[$t \geq 0$ —SS] [Inherited from GD:velVec —SS] [$v_{\text{launch}} > 0$ —SS] [MD:towardLauncher —SS] [$0 < \theta < \frac{\pi}{2}$ —SS] [MD:towardLauncher —SS]
[Notes —SS]	This theory starts with GD:velVec and switches to the magnitude/angle representation for the components of the initial velocity vector (MD:magAngleRep). That is, DD:speedIX and DD:speedIY are used for the initial velocity components. The magnitude of the initial velocity is called v_{launch}. The constraints on v_{launch} and θ are required to satisfy the assumption that the launcher must be aimed toward the target MD:towardLauncher for the coordinate system defined in PT:coordSyst.
Source	–
RefBy	PT:posVecInitPos

Context Theories Used by PT:velVecInitMagAndAngle

- CT:realArith

- CT:vectors
- CT:CartCoordSyst

Initial Theories Used by PT:velVecInitMagAndAngle

- GD:velVec
- PT:coordSyst
- DD:speedIX
- DD:speedIY

Preconditions for PT:velVecInitMagAndAngle

- A:timeStartZero (inherited from GD:velVec)
- MD:cartSyst (inherited from GD:velVec)
- A:twoD (inherited from GD:velVec)
- A:constAccelX (A:constAccel in x direction) (inherited from GD:velVec)
- A:constAccelY (A:constAccel in y direction) (inherited from GD:velVec)
- A:launchOrigin (inherited from PT:coordSyst)
- A:targetXAxis (inherited from PT:coordSyst)
- A:posXDirection (inherited from PT:coordSyst)
- A:yAxisGravity (inherited from PT:coordSyst)
- A:noObstruct (inherited from DD:speedIX and DD:SpeedIY)
- A:flatPlanet
- MD:magAngleRep
- MD:towardLauncher

Refname	**PT:posVecInitMagAndAngle**
Label	Position vector as a function of time for 2D projectile motion under constant acceleration in both the x and y directions using the magnitude and angle representation of the initial velocity.
Units	$\frac{\mathrm{m}}{\mathrm{s}}$
Equation	$$\mathbf{p}(t) = \begin{bmatrix} {p_{\text{x}}}^{\text{i}} + v_{\text{launch}} \cos(\theta) t + \frac{{a_{\text{x}}}^{\text{c}} t^2}{2} \\ {p_{\text{y}}}^{\text{i}} + v_{\text{launch}} \sin(\theta) t + \frac{{a_{\text{y}}}^{\text{c}} t^2}{2} \end{bmatrix}$$
Description	[$\mathbf{p}$: time $\rightarrow \mathbb{R}^2$ —SS] is the position [vector —SS] (m) [${p_{\text{x}}}^{\text{i}} : \mathbb{R}$ —SS] is the x-component of initial position (m) [${p_{\text{y}}}^{\text{i}} : \mathbb{R}$ —SS] is the y-component of initial position (m) [$v_{\text{launch}} : \mathbb{R}$ —SS] is the initial speed ($\frac{\mathrm{m}}{\mathrm{s}}$) [${a_{\text{x}}}^{\text{c}} : \mathbb{R}$ —SS] is the x-component of constant acceleration ($\frac{\mathrm{m}}{\mathrm{s}^2}$) [${a_{\text{y}}}^{\text{c}} : \mathbb{R}$ —SS] is the y-component of constant acceleration ($\frac{\mathrm{m}}{\mathrm{s}^2}$) [$t$: time —SS] is the time (s) $\theta : \mathbb{R}$ is the launch angle (rad)
[Constraints —SS]	[$t \geq 0$ —SS] [Inherited from GD:posVec —SS] [$v_{\text{launch}} > 0$ —SS] [MD:towardLauncher —SS] [$0 < \theta < \frac{\pi}{2}$ —SS] [MD:towardLauncher —SS]
[Notes —SS]	This theory starts with GD:posVec and switches to the magnitude/angle representation for the components of the initial velocity vector (MD:magAngleRep). That is, DD:speedIX and DD:speedIY are used for the initial velocity components. The magnitude of the initial velocity is called v_{launch}. The constraints on v_{launch} and θ are required to satisfy the assumption that the launcher must be aimed toward the target MD:towardLauncher for the coordinate system defined in PT:coordSyst.
Source	–
RefBy	PT:velVecPlanetaryGrav

Context Theories Used by PT:posVecInitMagAndAngle

- CT:realArith
- CT:vectors
- CT:CartCoordSyst

Initial Theories Used by PT:posVecInitMagAndAngle

- GD:posVec
- PT:coordSyst
- DD:speedIX
- DD:speedIY

Preconditions for PT:posVecInitMagAndAngle

- A:timeStartZero (inherited from GD:velVec)
- MD:cartSyst (inherited from GD:velVec)
- A:twoD (inherited from GD:velVec)
- A:constAccelX (A:constAccel in x direction) (inherited from GD:velVec)
- A:constAccelY (A:constAccel in y direction) (inherited from GD:velVec)
- A:launchOrigin (inherited from PT:coordSyst)
- A:targetXAxis (inherited from PT:coordSyst)
- A:posXDirection (inherited from PT:coordSyst)
- A:yAxisGravity (inherited from PT:coordSyst)
- A:noObstruct (inherited from DD:speedIX and DD:SpeedIY)
- A:flatPlanet
- MD:magAngleRep
- MD:towardLauncher

Refname	PT:posVecInitPos
Label	Position vector as a function of time for 2D projectile motion under constant acceleration in both the x and y directions using the magnitude and angle representation of the initial velocity with the initial position at the origin.
Units	$\frac{\mathrm{m}}{\mathrm{s}}$
Equation	$$\mathbf{p}(t) = \begin{bmatrix} v_{\text{launch}} \cos(\theta) t + \frac{a_{\text{x}}^{\text{c}} t^2}{2} \\ v_{\text{launch}} \sin(\theta) t + \frac{a_{\text{y}}^{\text{c}} t^2}{2} \end{bmatrix}$$
Description	[$\mathbf{p}$: time $\rightarrow \mathbb{R}^2$ —SS] is the position [vector —SS] (m) [$v_{\text{launch}} : \mathbb{R}$ —SS] is the initial speed ($\frac{\mathrm{m}}{\mathrm{s}}$) [${a_{\text{x}}}^{\text{c}} : \mathbb{R}$ —SS] is the x-component of constant acceleration ($\frac{\mathrm{m}}{\mathrm{s}^2}$) [${a_{\text{y}}}^{\text{c}} : \mathbb{R}$ —SS] is the y-component of constant acceleration ($\frac{\mathrm{m}}{\mathrm{s}^2}$) [$t$: time —SS] is the time (s) $\theta : \mathbb{R}$ is the launch angle (rad)
[Constraints —SS]	[$t \geq 0$ —SS] [Inherited from PT:posVecInitMagAndAngle —SS] [$v_{\text{launch}} > 0$ —SS] [Inherited from PT:posVecInitMagAndAngle —SS] [$0 < \theta < \frac{\pi}{2}$ —SS] [Inherited from PT:posVecInitMagAndAngle —SS]
[Notes —SS]	This theory starts with PT:posVecInitMagAndAngle and uses the location of the launcher at the origin (A:launchOrigin) to find the initial position in both coordinate directions is zero. The constraints on v_{launch} and θ are required to satisfy the assumption that the launcher must be aimed toward the target MD:towardLauncher for the coordinate system defined in PT:coordSyst.
Source	–
RefBy	PT:posVecPlanetaryGrav

Context Theories Used by PT:posVecInitPos

- CT:realArith

- CT:vectors
- CT:CartCoordSyst

Initial Theories Used by PT:posVecInitPos

- PT:posVecInitMagAndAngle

Preconditions for PT:posVecInitPos

- A:timeStartZero (inherited from GD:velVec)
- MD:cartSyst (inherited from GD:velVec)
- A:twoD (inherited from GD:velVec)
- A:constAccelX (A:constAccel in x direction) (inherited from GD:velVec)
- A:constAccelY (A:constAccel in y direction) (inherited from GD:velVec)
- A:launchOrigin (inherited from PT:coordSyst)
- A:targetXAxis (inherited from PT:coordSyst)
- A:posXDirection (inherited from PT:coordSyst)
- A:yAxisGravity (inherited from PT:coordSyst)
- A:noObstruct (inherited from PT:posVecInitMagAndAngle)
- A:flatPlanet
- MD:magAngleRep (inherited from PT:posVecInitMagAndAngle)
- MD:towardLauncher (inherited from PT:posVecInitMagAndAngle)

Refname	PT:velVecPlanetaryGrav
Label	Velocity vector as a function of time for 2D projectile motion under gravitational acceleration in both the y and zero acceleration in the x direction using the magnitude and angle representation of the initial velocity.
Units	$\frac{\text{m}}{\text{s}}$
Equation	$$\mathbf{v}(t) = \begin{bmatrix} v_{\text{launch}} \cos(\theta) \\ v_{\text{launch}} \sin(\theta) - gt \end{bmatrix}$$
Description	[$\mathbf{v}$: time $\rightarrow \mathbb{R}^2$ —SS] is the velocity [vector —SS] ($\frac{\text{m}}{\text{s}}$) [$v_{\text{launch}}$: $\mathbb{R}$ —SS] is the initial speed ($\frac{\text{m}}{\text{s}}$) [$t$: time —SS] is the time (s) [g : $\mathbb{R}$ —SS] is the constant acceleration due to gravity ($\frac{\text{m}}{\text{s}^2}$) θ : $\mathbb{R}$ is the launch angle (rad)
[Constraints —SS]	[$t \geq 0$ —SS] [Inherited from PT:velVecInitMagAndAngle —SS] [$v_{\text{launch}} > 0$ —SS] [Inherited from PT:velVecInitMagAndAngle —SS] [$0 < \theta < \frac{\pi}{2}$ —SS] [Inherited from PT:velVecInitMagAndAngle —SS]
[Notes —SS]	This theory starts with PT:velVecInitMagAndAngle and uses the constant acceleration for projectile motion on a planet (A:gravAccel). The x component of the acceleration is zero and the y component is $-g$. The negative sign is used because gravity acts in the opposite direction to the positive y direction assumed for the coordinate system defined in PT:coordSyst.
Source	–
RefBy	None

Context Theories Used by PT:velVecPlanetaryGrav

- CT:realArith

- CT:vectors
- CT:CartCoordSyst

Initial Theories Used by PT:velVecPlanetaryGrav

- PT:velVecInitMagAndAngle

Preconditions for PT:velVecPlanetaryGrav

- A:timeStartZero (inherited from GD:velVec)
- MD:cartSyst (inherited from GD:velVec)
- A:twoD (inherited from GD:velVec)
- A:accelXZero (the assumption A:constAccelX still applies, but the value for the constant is now set to 0)
- A:accelYGravity (the assumption A:constAccelY still applies, but the value for the constant acceleration is set to $-g$)
- A:launchOrigin (inherited from PT:coordSyst)
- A:targetXAxis (inherited from PT:coordSyst)
- A:posXDirection (inherited from PT:coordSyst)
- A:yAxisGravity (inherited from PT:coordSyst)
- A:noObstruct
- A:flatPlanet
- MD:magAngleRep (inherited from PT:posVecInitMagAndAngle)
- MD:towardLauncher (inherited from PT:posVecInitMagAndAngle)
- A:gravAccel

Refname	PT:posVecPlanetaryGrav
Label	Position vector as a function of time for 2D projectile motion under gravitational acceleration on a planet, with the x acceleration at 0 and the y direction acceleration at $-g$ using the magnitude and angle representation of the initial velocity with the initial position at the origin.
Units	$\frac{\text{m}}{\text{s}}$
Equation	$$\mathbf{p}(t) = \begin{bmatrix} v_{\text{launch}}\cos(\theta)t \\ v_{\text{launch}}\sin(\theta)t - \frac{gt^2}{2} \end{bmatrix}$$
Description	[$\mathbf{p}$: time $\rightarrow \mathbb{R}^2$ —SS] is the position [vector —SS] (m) [v_{launch} : $\mathbb{R}$ —SS] is the initial speed ($\frac{\text{m}}{\text{s}}$) [$g$: $\mathbb{R}$ —SS] is the gravitational acceleration ($\frac{\text{m}}{\text{s}^2}$) [$t$: time —SS] is the time (s) θ : $\mathbb{R}$ is the launch angle (rad)
[Constraints —SS]	[$t \geq 0$ —SS] [Inherited from PT:posVecInitPos —SS] [$v_{\text{launch}} > 0$ —SS] [Inherited from PT:posVecInitPos —SS] [$0 < \theta < \frac{\pi}{2}$ —SS] [Inherited from PT:posVecInitPos —SS]
[Notes —SS]	This theory starts with PT:posVecInitPos and uses the constant acceleration for projectile motion on a planet (A:gravAccel). The x component of the acceleration is zero and the y component is $-g$. The negative sign is used because gravity acts in the opposite direction to the positive y direction assumed for the coordinate system defined in PT:coordSyst.
Source	–
RefBy	IM:calOfLandingTime, IM:calOfLandingDist

Context Theories Used by PT:posVecPlanetaryGrav

- CT:realArith

- CT:vectors
- CT:CartCoordSyst

Initial Theories Used by PT:posVecPlanetaryGrav

- PT:posVecInitPos

Preconditions for PT:posVecPlanetaryGrav

- A:timeStartZero (inherited from GD:velVec)
- MD:cartSyst (inherited from GD:velVec)
- A:twoD (inherited from GD:velVec)
- A:constAccelX (A:constAccel in x direction) (inherited from GD:velVec)
- A:constAccelY (A:constAccel in y direction) (inherited from GD:velVec)
- A:launchOrigin (inherited from PT:coordSyst)
- A:targetXAxis (inherited from PT:coordSyst)
- A:posXDirection (inherited from PT:coordSyst)
- A:yAxisGravity (inherited from PT:coordSyst)
- A:noObstruct
- A:flatPlanet
- MD:magAngleRep (inherited from PT:posVecInitPos)
- MD:towardLauncher (inherited from PT:posVecInitPos)
- A:gravAccel

4.2.14 [Final Theories —SS]

This section transforms the problem defined in the problem description into one which is expressed in mathematical terms. It uses concrete symbols defined in the data definitions to replace the abstract symbols in the models identified in theoretical models and general definitions.

Refname	IM:calOfLandingTime
Label	Calculation of landing time
Input	v_{launch} [:ℝ —SS], θ [:ℝ —SS]
Output	t_{flight} [:ℝ —SS]
Input Constraints	$v_{\text{launch}} > 0$ $0 < \theta < \frac{\pi}{2}$
Output Constraints	$t_{\text{flight}} > 0$
Equation	$t_{\text{flight}} = \frac{2v_{\text{launch}} \sin(\theta)}{g}$
Description	[t_{flight} : ℝ —SS] is the flight duration (s) [v_{launch} : ℝ —SS] is the launch speed ($\frac{\text{m}}{\text{s}}$) [$\theta$: ℝ —SS] is the launch angle (rad) [g : ℝ —SS] is the magnitude of gravitational acceleration ($\frac{\text{m}}{\text{s}^2}$)
Notes	[The input constraints are inherited from PT:posVecPlanetaryGrav. The output constraint follows from the rules for multiplication and division of positive values, since $v_{\text{launch}} > 0$, $g > 0$, and $\sin\theta > 0$ for $0 < \theta < \frac{\pi}{2}$ (CT:realArith, CT:trigonometry). g is assumed to have the value defined in Sec:Values of Auxiliary Constants (A:gravAccelValue). —SS]
Source	–
RefBy	IM:calOfLandingDist

Detailed derivation of flight duration: From the y component of PT:posVecPlanetary-Grav we know:

$$p_{\text{y}} = v_{\text{launch}} \sin(\theta) t - \frac{g t^2}{2}$$

To find the time that the projectile lands, we want to find the t value (t_{flight}) where $p_{\text{y}} = 0$ (since the target is on the x-axis from A:targetXAxis). From the equation above we get:

$$v_{\text{launch}} \sin(\theta) t_{\text{flight}} - \frac{g {t_{\text{flight}}}^2}{2} = 0$$

Dividing by t_{flight} (with the constraint $t_{\text{flight}} > 0$ since time is greater than zero A:timeStartZero) gives us:

$$v_{\text{launch}} \sin(\theta) - \frac{g t_{\text{flight}}}{2} = 0$$

Solving for t_{flight} gives us:

$$t_{\text{flight}} = \frac{2 v_{\text{launch}} \sin(\theta)}{g}$$

Context Theories Used by IM:calOfLandingTime

- CT:realArith
- CT:trigonometry

Initial Theories Used by IM:calOfLandingTime

- PT:posVecPlanetaryGrav

Preconditions for IM:calOfLandingTime

- A:timeStartZero (inherited from PT:posVecPlanetaryGrav)
- MD:cartSyst (inherited from PT:posVecPlanetaryGrav)
- A:twoD (inherited from PT:posVecPlanetaryGrav)
- A:accelXZero (inherited from PT:posVecPlanetaryGrav)
- A:accelYGravity (inherited from PT:posVecPlanetaryGrav)
- A:launchOrigin (inherited from PT:posVecPlanetaryGrav)
- A:targetXAxis (inherited from PT:posVecPlanetaryGrav)
- A:posXDirection (inherited from PT:posVecPlanetaryGrav)

- A:yAxisGravity (inherited from PT:posVecPlanetaryGrav)
- A:noObstruct
- A:flatPlanet
- MD:magAngleRep (inherited from PT:posVecPlanetaryGrav)
- MD:towardLauncher (inherited from PT:posVecPlanetaryGrav)
- A:gravAccelValue (replaces A:gravAccel because a specific value for the gravitational acceleration has been selected.)

Refname	IM:calOfLandingDist
Label	Calculation of landing position
Input	$v_{\text{launch}} : \mathbb{R}$, $\theta : \mathbb{R}$
Output	p_{land} [: $\mathbb{R}$ —SS]
Input Constraints	$$v_{\text{launch}} > 0$$ $$0 < \theta < \frac{\pi}{2}$$
Output Constraints	$$p_{\text{land}} > 0$$
Equation	$$p_{\text{land}} = \frac{2{v_{\text{launch}}}^2 \sin(\theta) \cos(\theta)}{g}$$
Description	$p_{\text{land}} : \mathbb{R}$ is the landing position (m) $v_{\text{launch}} : \mathbb{R}$ is the launch speed ($\frac{\text{m}}{\text{s}}$) $\theta : \mathbb{R}$ is the launch angle (rad) $g : \mathbb{R}$ is the magnitude of gravitational acceleration ($\frac{\text{m}}{\text{s}^2}$)
Notes	[The input constraints are inherited from IM:calOfLandingTime. The output constraint follows from the rules for multiplication and division of positive values, since $v_{\text{launch}} > 0$, $g > 0$, $\sin\theta > 0$ for $0 < \theta < \frac{\pi}{2}$ and $\cos\theta > 0$ for $0 < \theta < \frac{\pi}{2}$ (CT:realArith, CT:trigonometry). g is assumed to have the value defined in Sec:Values of Auxiliary Constants (A:gravAccelValue). —SS] [A detailed derivation of the Equation is provided below. —SS]
Source	–
RefBy	IM:offsetIM and FR:Calculate-Values

Detailed derivation of landing position: From the x component of PT:posVecPlanetaryGrav we know:

$$p_{\mathrm{x}} = v_{\mathrm{launch}} \cos(\theta) t$$

To find the landing position, we want to find the p_{x} value (p_{land}) at flight duration [t_{flight}—SS] (from IM:calOfLandingTime):

$$p_{\mathrm{land}} = \frac{v_{\mathrm{launch}} \cos(\theta) \cdot 2 v_{\mathrm{launch}} \sin(\theta)}{g}$$

Rearranging this gives us the required equation:

$$p_{\mathrm{land}} = \frac{2 {v_{\mathrm{launch}}}^2 \sin(\theta) \cos(\theta)}{g}$$

Context Theories Used by IM:calOfLandingDist

- CT:realArith
- CT:trigonometry

Initial Theories Used by IM:calOfLandingDist

- PT:posVecPlanetaryGrav
- IM:calOfLandingTime

Preconditions for IM:calOfLandingDist

- A:timeStartZero (inherited from PT:posVecPlanetaryGrav
- MD:cartSyst (inherited from PT:posVecPlanetaryGrav)
- A:twoD (inherited from PT:posVecPlanetaryGrav)
- A:accelXZero (inherited from PT:posVecPlanetaryGrav)
- A:accelYGravity (inherited from PT:posVecPlanetaryGrav)
- A:launchOrigin (inherited from PT:posVecPlanetaryGrav)
- A:targetXAxis (inherited from PT:posVecPlanetaryGrav)
- A:posXDirection (inherited from PT:posVecPlanetaryGrav)
- A:yAxisGravity (inherited from PT:posVecPlanetaryGrav)

- A:noObstruct
- A:flatPlanet
- MD:magAngleRep (inherited from PT:posVecPlanetaryGrav)
- MD:towardLauncher (inherited from PT:posVecPlanetaryGrav)
- A:gravAccelValue (inherited from PT:posVecPlanetaryGrav)

Refname	**IM:offsetIM**
Label	Offset
Input	$p_{\text{land}} : \mathbb{R}$, $p_{\text{target}} : \mathbb{R}$
Output	$d_{\text{offset}} : \mathbb{R}$
Input Constraints	$$p_{\text{land}} > 0$$ $$p_{\text{target}} > 0$$
Output Constraints	[$$d_{\text{offset}} > -p_{\text{target}}$$ —SS]
Equation	$$d_{\text{offset}} = p_{\text{land}} - p_{\text{target}}$$
Description	$d_{\text{offset}} : \mathbb{R}$ is the distance between the target position and the landing position (m) $p_{\text{land}} : \mathbb{R}$ is the landing position (m) $p_{\text{target}} : \mathbb{R}$ is the target position (m)
Notes	p_{land} is from IM:calOfLandingDist. The constraints $p_{\text{land}} > 0$ comes from IM:calOfLandingDist and the constraint $p_{\text{target}} > 0$ comes from A:posXDirection. [The constraint $d_{\text{offset}} > -p_{\text{target}}$ is from the fact that there is a lower bound of zero on p_{land}. —SS]
Source	–
RefBy	IM:messageIM, FR:Output-Values, and FR:Calculate-Values

Context Theories Used by IM:offsetIM

- CT:realArith

Initial Theories Used by IM:offsetIM

- IM:calOfLandingDist

Preconditions for IM:offsetIM

- A:timeStartZero (inherited from IM:calOfLandingDist)
- MD:cartSyst (inherited from IM:calOfLandingDist)
- A:twoD (inherited from IM:calOfLandingDist)
- A:accelXZero (inherited from IM:calOfLandingDist)
- A:accelYGravity (inherited from IM:calOfLandingDist)
- A:launchOrigin (inherited from IM:calOfLandingDist)
- A:targetXAxis (inherited from IM:calOfLandingDist)
- A:posXDirection (inherited from IM:calOfLandingDist)
- A:yAxisGravity (inherited from IM:calOfLandingDist)
- A:noObstruct
- A:flatPlanet
- MD:magAngleRep (inherited from IM:calOfLandingDist)
- MD:towardLauncher (inherited from IM:calOfLandingDist)
- A:gravAccelValue (inherited from IM:calOfLandingDist)

Refname	IM:messageIM
Label	Output message
Input	$d_{\text{offset}} : \mathbb{R}$, $p_{\text{target}} : \mathbb{R}$
Output	s : string
Input Constraints	$$d_{\text{offset}} > -p_{\text{target}}$$ $$p_{\text{target}} > 0$$
Output Constraints	[None —SS]
Equation	$$s = \begin{cases} \text{“The target was hit.”}, & \lvert\frac{d_{\text{offset}}}{p_{\text{target}}}\rvert < \varepsilon \\ \text{“The projectile fell short.”}, & d_{\text{offset}} < 0 \\ \text{“The projectile went long.”}, & d_{\text{offset}} > 0 \end{cases}$$
Description	s is the output message as a string (Unitless) d_{offset} is the distance between the target position and the landing position (m) p_{target} is the target position (m) ε is the hit tolerance (Unitless)
Notes	d_{offset} is from IM:offsetIM. The constraints $p_{\text{target}} > 0$, [and $d_{\text{offset}} > -p_{\text{target}}$ —SS], are from [IM:offsetIM —SS]. ε is defined in Sec:Values of Auxiliary Constants.
Source	–
RefBy	FR:Output-Values and FR:Calculate-Values

Context Theories Used by IM:messageIM

- CT:realArith

Initial Theories Used by IM:messageIM

- IM:offsetIM

Preconditions for IM:messageIM

- A:timeStartZero (inherited from IM:offsetIM)
- MD:cartSyst (inherited from IM:offsetIM)
- A:twoD (inherited from IM:offsetIM)
- A:accelXZero (inherited from IM:offsetIM)
- A:accelYGravity (inherited from IM:offsetIM)
- A:launchOrigin (inherited from IM:offsetIM)
- A:targetXAxis (inherited from IM:offsetIM)
- A:posXDirection (inherited from IM:offsetIM)
- A:yAxisGravity (inherited from IM:offsetIM)
- A:noObstruct
- A:flatPlanet
- MD:magAngleRep (inherited from IM:offsetIM)
- MD:towardLauncher (inherited from IM:offsetIM)
- A:gravAccelValue (inherited from IM:offsetIM)

4.2.15 [Rationale Theories —SS]

The theories in this section are not part of the final theories. Their purpose is to express the theories that are used to justify and support that rationale for the assumptions.

Refname	RT:lngDstErr
Label	[Error introduced by assumming a flat planet instead of accounting for curvature. —SS]
Units	m
Equation	$$\Delta = \sqrt{a^2 \left(1 - \cos\left[\frac{L}{a}\right]\right)^2 + \left(L - a\sin\left[\frac{L}{a}\right]\right)^2}$$
Description	[Δ : $\mathbb{R}$ —SS] is the distance between the projectile's location via a flat planet model versus a curved planet model. (m) [a : $\mathbb{R}$ —SS] is semimajor axis length for an oblate spheroid for the equipotential gravitational surface of a planet. (m) [L : $\mathbb{R}$ —SS] is distance covered by the flight. (m)
[Constraints —SS]	[None —SS]
Source	[[4] —SS]
RefBy	Rationale section for final theories

Context Theories Used by RT:lngDstErr

- CT:realArith
- CT:CartCoordSyst
- CT:vectors

4.2.16 Data Constraints

The Data Constraints Table shows the data constraints on the input variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable. The uncertainty column provides an estimate of the confidence with which the physical quantities can be measured. This information would be part of the input if one were performing an uncertainty quantification exercise. The constraints are conservative to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario.

Table 4: Input Data Constraints

Var	Physical Constraints	Typical Value	Uncert.
p_{target}	$p_{\text{target}} > 0$ [(IM:messageIM) —SS]	1000 m	10%
v_{launch}	$v_{\text{launch}} > 0$ [(IM:calOfLandingDist) —SS]	100 $\frac{\text{m}}{\text{s}}$	10%
θ	$0 < \theta < \frac{\pi}{2}$ [(IM:calOfLandingDist) —SS]	$\frac{\pi}{4}$ rad	10%

4.2.17 Properties of a Correct Solution

The Data Constraints Table shows the data constraints on the output variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable.

Table 5: Output Data Constraints

Var	Physical Constraints
p_{land}	$p_{\text{land}} > 0$ [(IM:calOfLandingDist) —SS]
d_{offset}	$d_{\text{offset}} > -p_{\text{target}}$ [(IM:offsetIM) —SS]
t_{flight}	$t_{\text{flight}} > 0$ [(IM:calOfLandingTime) —SS]

5 Requirements

This section provides the functional requirements, the tasks and behaviours that the software is expected to complete, and the non-functional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

This section provides the functional requirements, the tasks and behaviours that the software is expected to complete.

Input-Values: Input the values from Tab:ReqInputs.

Verify-Input-Values: Check the entered input values to ensure that they do not exceed the data constraints. If any of the input values are out of bounds, an error message is displayed and the calculations stop.

Calculate-Values: Calculate the following values: t_{flight} (from IM:calOfLandingTime), p_{land} (from IM:calOfLandingDist), d_{offset} (from IM:offsetIM), and s (from IM:messageIM).

Output-Values: Output t_{flight} (from IM:calOfLandingTime), s (from IM:messageIM), and d_{offset} (from IM:offsetIM).

Table 6: Required Inputs following FR:Input-Values

Symbol	Description	Units
p_{target}	Target position	m
v_{launch}	Launch speed	$\frac{\text{m}}{\text{s}}$
θ	Launch angle	rad

5.2 Non-Functional Requirements

This section provides the non-functional requirements, the qualities that the software is expected to exhibit.

Correct: The outputs of the code have the properties described in Properties of a Correct Solution.

Verifiable: The code is tested with complete verification and validation plan.

Understandable: The code is modularized with complete module guide and module interface specification.

Reusable: The code is modularized.

Maintainable: The traceability between requirements, assumptions, theoretical models, general definitions, data definitions, instance models, likely changes, unlikely changes, and modules is completely recorded in traceability matrices in the SRS and module guide.

Portable: The code is able to be run in different environments.

5.3 [Rationale —SS]

[Capture the rationale for the scope assumptions and final theory assumptions. The rationale could vary between problems. For instance, for projectile motion the rationale could be that it is being used for teaching purposes. If the theories are used to solve an actual science or engineering problem, the rationale would need more justification. —SS] [Justify the scope decisions. No need to justify modelling decisions. —SS] [Should requirements be added related to guaranteeing assumptions and constraints? (As is done after a hazard analysis.) Requirements could be added to check the input constraints, like $x > 0$. Requirements could be added to check neglecting curvature. —SS]

5.3.1 [Rationale for Scope Decisions —SS]

[The rationale for the scope decisions. This may require introducing new assumptions. —SS]

- SD:noOrient [Since the software is in the context of education and exploration (Section 3.1), introducing rotational motion would make the problem too complex. To keep things simple we are only interested in the coordinates of the centre of mass through the flight (A:centreMass). —SS]

- SD:kinOnly [Since the software is in the context of education and exploration (Section 3.1), introducing forces makes the problem too complex. This means we are neglecting the force of air drag (A:neglectDrag) —SS]

5.3.2 [Rationale for Modelling Decisions —SS]

[The rationale for the modelling decisions. —SS]

- MD:cartSyst [A Cartesian coordinate system is used because the theories focus on rectilinear motion. If the curvature of the planet is considered, then a spherical coordinate system would be more natural. —SS]

- yAxisGravity [The standard convention is used of labelling the vertical direction y and making up positive. —SS]

- launchOrigin [Placing the launcher at the origin simplifies the equations because the initial coordinates for the position are zero. —SS]

- targetXAxis [Placing the target on the x axis makes the calculations easier because the y coordinate for the target is zero. —SS]

- posXDirection [The standard convention is used of making the horizontal direction to the right positive. —SS]

- timeStartZero [The equations are simplified if the initial time is assumed to be zero. —SS]

- MD:towardLauncher [The equations are simplified if the cases that would never hit the target are eliminated. —SS]

- MD:magAngleRep [This is a standard representation for projectile problems. —SS]

5.3.3 [Rationale for Final Theory Assumptions —SS]

[The rationale for the final theory preconditions. The relevant assumptions needed for the final theories will have propagated up from all intermediate theories that are used for the final theory. —SS]

[From IM:messageIM, removing modelling decisions: —SS]

- A:twoD [We can consider the problem as two dimensional because there are no forces acting on the projectile to change its course. We also need to neglect the rotation of the planet (Coriolis effect) (A:negPlanetRot —SS].

- A:accelXZero [The projectile motion problem is on a planet, like Earth. There is nothing else as massive as the planet to create a gravitational force of attraction on the projectile. Therefore, the only relevant gravitational force is the one straight down; there is no horizonal gravitational force. —SS]
- A:accelYGravity [The constant acceleration in the y direction is the attraction from the massive planet to the projectile. The gravitational attraction depends on Newton's law of gravitational attraction (not shown in this document). The attraction varies with distance, given that the initial velocities are relatively small, the projectile will not get very high and the acceleration due to gravity will only have a negligible change over the projectile's flight. —SS]
- A:gravAccelValue [The usual value of acceleration on Earth at sea level is assummed. —SS]
- A:noObstruct [If there are obstructions then there is no reason to solve the projectile motion problem. —SS]
- A:flatPlanet [The curvature of the planet is large enough that we can assume the planet is flat. This is only possible in the scope of problems with velocities and distances that are relatively small compared to ballistics problems (Section 3.1). The error can be calculated using RT:lngDstErr. For the Earth $a = 6378137.0$m [4]. For the typical values $L = 1019.4$m. With these numbers the error $\Delta = 0.8$ m. The relative error is 0.8/1019.4 or 0.008 %. —SS]

5.3.4 [Rationale for Typical Values —SS]

[The typical values for the inputs in Table 4.2.16 are based on the scope of short flight distance and relatively small initial magnitude of velocity as compared to ballistics problems. The initial velocity (v_{launch}) of 100 $\frac{\text{m}}{\text{s}}$ converts to 100/1000*60*60 = 360 $\frac{\text{km}}{\text{hr}}$. The speed of a bullet, on the other hand, is over 1000 $\frac{\text{km}}{\text{hr}}$. —SS]

[Using the typical values p_{land} is calculated as 1019.4 m, which is close to the given value for the target. A typical value that gets close to the target is logical, assumming that the user has a good feel for the projectile's motion based on previous experiences. —SS]

[We might want to change the typical values to be slower and over shorter distances. Numbers more like those from throwing or hitting a ball in a typical sport might be more typical when the scope is outside the range of ballistics. —SS]

6 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" should be modified as well. Tab:TraceMatAvsA shows the dependencies of the assumptions on each other.

Tab:TraceMatAvsAll shows the dependencies of the data definitions, theoretical models, general definitions, instance models, requirements, likely changes, and unlikely changes on the assumptions. Tab:TraceMatRefvsRef shows the dependencies of the data definitions, theoretical models, general definitions, and instance models on each other. Tab:TraceMatAllvsR shows the dependencies of the requirements and goal statements on the data definitions, theoretical models, general definitions, and instance models.

Tabl

	A:twoD	**MD:cartSyst**	**A:yAxisGravity**	**A:launchOrigin**	**A:targetXAxi**
A:twoD					
A:cartSyst					
A:yAxisGravity					
A:launchOrigin					
A:targetXAxis					
A:posXDirection					
A:constAccel					
A:accelXZero					
A:accelYGravity			X		
A:neglectDrag					
A:centreMass					
A:noObstruct					
A:neglectCurv					
A:timeStartZero					
A:gravAccelValue					

	A:twoD	**A:cartSyst**	**A:yAxisGravity**	**A:launchOrigin**	**A:targetXA**
DD:vecMag					
DD:speedIX					
DD:speedIY					
TM:acceleration					
TM:velocity					

Table

	A:twoD	A:cartSyst	A:yAxisGravity	A:launchOrigin	A:targetXA
GD:rectVel					
GD:rectPos					
GD:velVec	X	X			
GD:posVec	X	X			
IM:calOfLandingTime			X	X	X
IM:calOfLandingDist			X	X	
IM:offsetIM					
IM:messageIM					
FR:Input-Values					
FR:Verify-Input-Values					
FR:Calculate-Values					
FR:Output-Values					
NFR:Correct					
NFR:Verifiable					
NFR:Understandable					
NFR:Reusable					
NFR:Maintainable					
NFR:Portable					

Table 9: Traceabilit

	DD:vecMag	DD:speedIX	DD:speedIY	TM:acceleration	TM:veloc
DD:vecMag					
DD:speedIX	X				
DD:speedIY	X				
TM:acceleration					
TM:velocity					
GD:rectVel				X	
GD:rectPos					X
GD:velVec					

Table 9: Traceability Matr

	DD:vecMag	DD:speedIX	DD:speedIY	TM:acceleration	TM:veloc
GD:posVec					
IM:calOfLandingTime			X		
IM:calOfLandingDist		X			
IM:offsetIM					
IM:messageIM					

	DD:vecMag	DD:speedIX	DD:speedIY	TM:acceleration	TM:velo
GS:targetHit					
FR:Input-Values					
FR:Verify-Input-Values					
FR:Calculate-Values					
FR:Output-Values					
NFR:Correct					
NFR:Verifiable					
NFR:Understandable					
NFR:Reusable					
NFR:Maintainable					
NFR:Portable					

The purpose of the traceability graphs is also to provide easy references on what has to be additionally modified if a certain component is changed. The arrows in the graphs represent dependencies. The component at the tail of an arrow is depended on by the component at the head of that arrow. Therefore, if a component is changed, the components that it points to should also be changed. Fig:TraceGraphAvsA shows the dependencies of assumptions on each other. Fig:TraceGraphAvsAll shows the dependencies of data definitions, theoretical models, general definitions, instance models, requirements, likely changes, and unlikely changes on the assumptions. Fig:TraceGraphRefvsRef shows the dependencies of data definitions, theoretical models, general definitions, and instance models on each other. Fig:TraceGraphAllvsR shows the dependencies of requirements and goal statements on the data definitions, theoretical models, general definitions, and instance models. Fig:TraceGraphAllvsAll shows the dependencies of dependencies of assumptions, models, definitions, requirements, goals, and changes with each other.

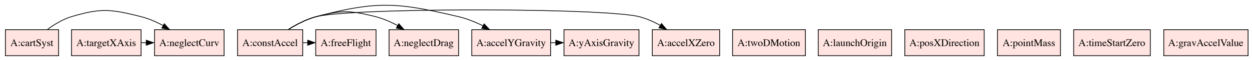

Figure 3: TraceGraphAvsA

Figure 4: TraceGraphAvsAll

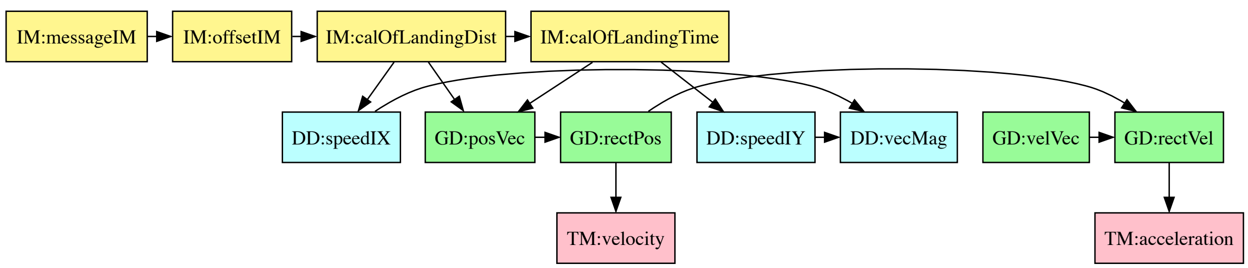

Figure 5: TraceGraphRefvsRef

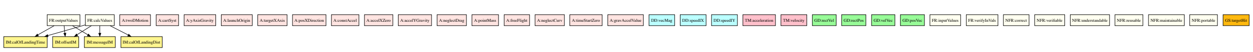

Figure 6: TraceGraphAllvsR

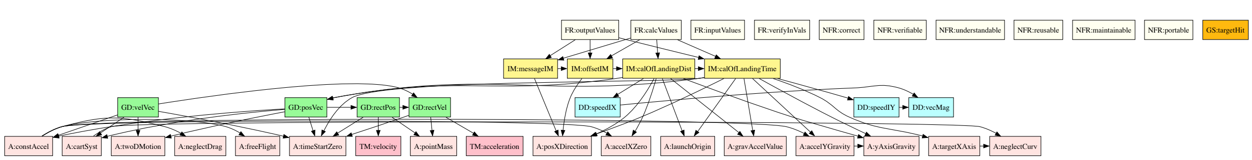

Figure 7: TraceGraphAllvsAll

For convenience, the following graphs can be found at the links below:

- TraceGraphAvsA
- TraceGraphAvsAll
- TraceGraphRefvsRef
- TraceGraphAllvsR
- TraceGraphAllvsAll

7 Values of Auxiliary Constants

This section contains the standard values that are used for calculations in Projectile.

Table 11: Auxiliary Constants

Symbol	Description	Value	Unit
g	magnitude of gravitational acceleration	9.8	$\frac{\text{m}}{\text{s}^2}$
ε	hit tolerance	2.0%	–
π	ratio of circumference to diameter for any circle	3.14159265	–

[The relationship between the theories, assumptions and modelling decisions is summarized in Figure ??. —SS]

8 References

[1] Wikipedia Contributors. *Acceleration*. https://en.wikipedia.org/wiki/Acceleration. June 2019.

[2] Wikipedia Contributors. *Cartesian coordinate system*. https://en.wikipedia.org/wiki/Cartesian_coordinate_system. June 2019.

[3] Wikipedia Contributors. *Velocity*. https://en.wikipedia.org/wiki/Velocity. June 2019.

[4] David Crouse. "Basic tracking using nonlinear 3D monostatic and bistatic measurements". In: *IEEE Aerospace and Electronic Systems Magazine* 29.8 (2014), pp. 4–53. DOI: 10.1109/MAES.2014.120229.

[5] R. C. Hibbeler. *Engineering Mechanics: Dynamics*. Pearson Prentice Hall, 2004.

Figure 8: Relationship between theories, assumptions and modelling decisions for Projectile.